

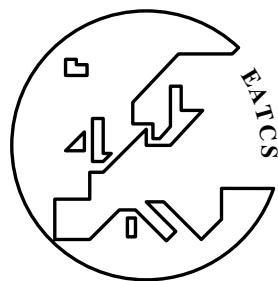
ISSN 0252-9742

Bulletin

of the

European Association for Theoretical Computer Science

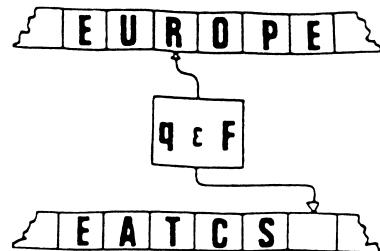
EATCS



Number 144

October 2024

**COUNCIL OF THE
EUROPEAN ASSOCIATION FOR
THEORETICAL COMPUTER SCIENCE**



PRESIDENT:	GIUSEPPE F. ITALIANO	ITALY
VICE PRESIDENTS:	ANTOINE AMARILLI INGE LI GØRTZ	FRANCE DENMARK
TREASURER:	GABRIELE FICI	ITALY
BULLETIN EDITOR:	STEFAN SCHMID	GERMANY

ANTOINE AMARILLI	FRANCE	THORE HUSFELDT	SWEDEN, DENMARK
IVONA BEZAKOVA	USA	GIUSEPPE F. ITALIANO	ITALY
TIZIANA CALAMONERI	ITALY	EMANUELA MERELLI	ITALY
THOMAS COLCOMBET	FRANCE	ANCA MUSCHOLL	FRANCE
ARTUR CZUMAJ	UK	CHARLES PAPERMAN	FRANCE
ANNE DRIEMEL	GERMANY	EVA ROTENBERG	DENMARK
FUNDA ERGÜN	USA	JIRI SGALL	CZECH REPUBLIC
JAVIER ESPARZA	GERMANY	JUKKA SUOMELA	FINLAND
GABRIELE FICI	ITALY	SZYMON TORUNCZYK	POLAND
INGE LI GOERTZ	DENMARK	BIANCA TRUTHE	GERMANY
FABRIZIO GRANDONI	SWITZERLAND		

PAST PRESIDENTS:

MAURICE NIVAT	(1972–1977)	MIKE PATERSON	(1977–1979)
ARTO SALOMAA	(1979–1985)	GRZEGORZ ROZENBERG	(1985–1994)
WILFRED BRAUER	(1994–1997)	JOSEP DÍAZ	(1997–2002)
MOGENS NIELSEN	(2002–2006)	GIORGIO AUSIELLO	(2006–2009)
BURKHARD MONIEN	(2009–2012)	LUCA ACETO	(2012–2016)
PAUL SPIRAKIS	(2016–2020)	ARTUR CZUMAJ	(2020–2024)

SECRETARY OFFICE:	DMITRY CHISTIKOV EFI CHITA	UK GREECE
-------------------	-------------------------------	--------------

EATCS COUNCIL MEMBERS

EMAIL ADDRESSES

ANTOINE AMARILLI	A3NM@A3NM.NET
IVONA BEZAKOVA	IB@CS.RIT.EDU
TIZIANA CALAMONERI	CALAMO@DI.UNIROMA1.IT
THOMAS COLCOMBET	THOMAS.COLCOMBET@IRIF.FR
ARTUR CZUMAJ	A.CZUMAJ@WARWICK.AC.UK
ANNE DRIEMEL	DRIEMEL@CS.UNI-BONN.DE
FUNDA ERGÜN	CHAIR.SIGACT@SIGACT.ACM.ORG
JAVIER ESPARZA	ESPARZA@IN.TUM.DE
GABRIELE FICI	GABRIELE.FICI@UNIPA.IT
INGE LI GOERTZ	INGE@DTU.DK
FABRIZIO GRANDONI	FABRIZIO@IDSIA.CH
THORE HUSFELDT	THORE@ITU.DK
GIUSEPPE F. ITALIANO	GIUSEPPE.ITALIANO@UNIROMA2.IT
EMANUELA MERELLI	EMANUELA.MERELLI@UNICAM.IT
ANCA MUSCHOLL	ANCA@LABRI.FR
CHARLES PAPERMAN	CHARLES.PAPERMAN@UNIV-LILLE.FR
EVA ROTENBERG	EVA@ROTHENBERG.DK
STEFAN SCHMID	STEFAN.SCHMID@TU-BERLIN.DE
JIRI SGALL	SGALL@IUUK.MFF.CUNI.CZ
JUKKA SUOMELA	JUKKA.SUOMELA@AALTO.FI
SZYMON TORUNCZYK	SZYMTOR@MIMUW.EDU.PL
BIANCA TRUTHE	BIANCA.TRUTHE@INFORMATIK.UNI-GIESSEN.DE

Bulletin Editor: Stefan Schmid, Berlin, Germany
Cartoons: DADARA, Amsterdam, The Netherlands

The bulletin is entirely typeset by PDF_TE_X and Con_TE_XT in TXFONTS.

All contributions are to be sent electronically to

bulletin@eatcs.org

and must be prepared in L_AT_EX₂_E using the class beatcs.cls (a version of the standard L_AT_EX₂_E article class). All sources, including figures, and a reference PDF version must be bundled in a ZIP file.

Pictures are accepted in EPS, JPG, PNG, TIFF, MOV or, preferably, in PDF. Photographic reports from conferences must be arranged in ZIP files layed out according to the format described at the Bulletin's web site. Please, consult <http://www.eatcs.org/bulletin/howToSubmit.html>.

We regret we are unfortunately not able to accept submissions in other formats, or indeed submission not *strictly* adhering to the page and font layout set out in beatcs.cls. We shall also not be able to include contributions not typeset at camera-ready quality.

The details can be found at <http://www.eatcs.org/bulletin>, including class files, their documentation, and guidelines to deal with things such as pictures and overfull boxes. When in doubt, email bulletin@eatcs.org.

Deadlines for submissions of reports are January, May and September 15th, respectively for the February, June and October issues. Editorial decisions about submitted technical contributions will normally be made in 6/8 weeks. Accepted papers will appear in print as soon as possible thereafter.

The Editor welcomes proposals for surveys, tutorials, and thematic issues of the Bulletin dedicated to currently hot topics, as well as suggestions for new regular sections.

The EATCS home page is <http://www.eatcs.org>

Table of Contents

EATCS MATTERS

LETTER FROM THE PRESIDENT	3
LETTER FROM THE EDITOR	5

EATCS COLUMNS

THE INTERVIEW COLUMN, *by C. Avin, S. Schmid*

KNOW THE PERSON BEHIND THE PAPERS TODAY: BRUCE MAGGS	11
---	----

THE VIEWPOINT COLUMN, *by S. Schmid*

THE ROLE OF AN INVITED SPEAKER AT A CONFERENCE, <i>by L. Aceto</i>	19
---	----

TCS ON THE WEB, *by S. Neumann*

PROMOTING THEORETICAL COMPUTER SCIENCE ON SOCIAL MEDIA, A CONVERSATION WITH CLÉMENT CANONNE, <i>by S. Neumann</i>	25
---	----

THE DISTRIBUTED COMPUTING COLUMN, *by S. Gilbert*

MINIMAL REQUIREMENTS FOR BIT-DISSEMINATION WITH PASSIVE COMMUNICATION, <i>by R. Vacus</i>	31
--	----

THE COMPUTATIONAL COMPLEXITY COLUMN, *by M. Koucký*

SEVEN WAYS OF DECIDING WHETHER MOST VERTEX SETS COVER A GRAPH, <i>by T. Tantau</i>	45
---	----

THE EDUCATION COLUMN *by D. Komm and T. Zeume*

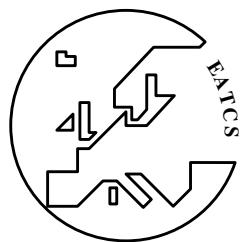
TEACHING DESIGN OF ALGORITHMS IN HIGH SCHOOLS BY CONSTRUCTIVE INDUCTION, <i>by J. Hromkovic, R. Lacher</i> ...	105
---	-----

THE LOGIC IN COMPUTER SCIENCE COLUMN, *by Y. Gurevich*

ON A MEASURE OF INTELLIGENCE, <i>by Y. Gurevich</i>	121
---	-----

EATCS LEAFLET	134
---------------------	-----

EATCS Matters





Dear EATCS members,

This is my first letter as President of EATCS, and I feel excited about it. It is an honor and a privilege to serve our community in this new role, and I am committed to working tirelessly for EATCS to advance our field of theoretical computer science. I would like to start by thanking Artur Czumaj for his exemplary leadership as previous President. Following in his footsteps will undoubtedly be a challenge, but I am fortunate to benefit from the groundwork he has laid!

We are living in a particularly exciting time for the recognition of computer science. The recent Physics Nobel Prize 2024 awarded to John J. Hopfield and Geoffrey E. Hinton "for foundational discoveries and inventions that enable machine learning with artificial neural networks", and the Chemistry Nobel Award 2024 awarded to Demis Hassabis and John M. Jumper for their work on "protein structure prediction" with the deep learning model AlphaFold2, highlight the immense impact of machine learning on the world. It is also a very important time for the recognition of theoretical computer science, with the 2023 Turing Award being given to Awi Wigderson "for foundational contributions to the theory of computation, including reshaping our understanding of the role of randomness in computation and mathematics, and for his decades of intellectual leadership in theoretical computer science".

On a sadder note, I can't help but think about the untimely passing of Luca



Trevisan. Luca was a colleague and a friend, and I believe that the EATCS community has experienced a great loss. His impact on theoretical computer science is invaluable, and I am sure that his memory will continue to inspire us all. To commemorate Luca's significant contributions to the field, we are working with the Computational Complexity Conference to have their best student paper award named after him.

In a few months we will have calls for nominations for many awards, including the EATCS Award, the Presburger Award, the EATCS Distinguished Dissertation Award, the EATCS Fellows, and some other joint awards, such as the Gödel Prize, the Alonzo Church Award, and the Dijkstra Prize. Let me take the opportunity to thank all the Award Committee Members for their important service to our community. Please consider sending your strong nominations for all those awards!

As we move forward, I am eager to hear your thoughts and ideas on how we can best serve our EATCS community. Please do not hesitate to reach out to me with any feedback or suggestions. Together, we can continue to push the boundaries of theoretical computer science and make an impact for a better world!

*Giuseppe F. Italiano
Luiss University, Rome
President of EATCS
president@eatcs.org*

October 2024



Dear EATCS member!

This issue of the Bulletin comes with several innovations.

First of all, I am very happy to welcome a new editor, Stefan Neumann from TU Wien, Austria, who will be in charge of a new column, "TCS on the Web". In his first column, Stefan Neumann leads a conversation with Clement Canonne, one of the most important theoretical computer science influencers. I am also very happy to welcome Thomas Zeume from the Ruhr-University Bochum, Germany. He will replace Juraj Hromkovic as editor of the Education Column, and will be responsible for it together with Dennis Komm. On this occasion, I would like to thank Juraj Hromkovic very much for all his contributions to the Bulletin over all these years. The Education Column in this issue will present an algorithms design strategy called "constructive induction" that enables high school students to solve a large variety of algorithmic problems. The Interview Column features Bruce Maggs who shares with us what influenced him in his career, and provides perspectives and advice how our community can evolve. In the Viewpoint Column, our former president Luca Aceto reflects on the role of an invited speaker at a conference. The Distributed Computing Column, edited by Seth Gilbert, features Robin Vacus, winner of the 2024 Principles of Distributed Computing Doctoral Dissertation Award. His thesis on "Algorithmic Perspectives to Collective Natural Phenomena" explores how



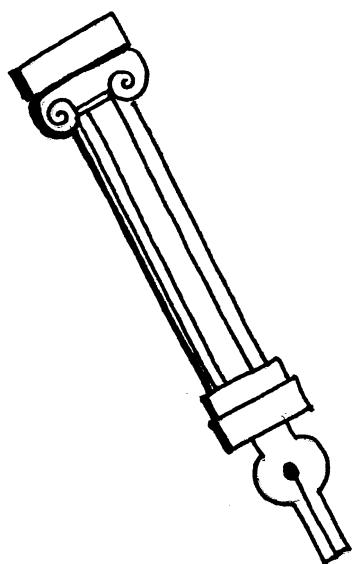
distributed computing techniques can help to understand biological processes found in the real world. It examines a variety of different phenomena. The Computational Complexity Column, edited by Michal Koucký, features an article by Till Tantau who explores and discusses seven ways of deciding whether most vertex sets cover a graph. In the Logic in Computer Science Column, Yuri Gurevich discusses intelligence, measuring intelligence, and related issues, inspired by the article “On the measure of intelligence” by François Chollet.

Enjoy the new Bulletin and I wish you a wonderful Autumn!

*Stefan Schmid, Berlin
October 2024*

Institutional Sponsors

EATCS
Columns



The Bulletin of the EATCS

THE INTERVIEW COLUMN

BY

CHEN AVIN AND STEFAN SCHMID

Ben Gurion University, Israel and TU Berlin, Germany
`{chenavin, schmiste}@gmail.com`

KNOW THE PERSON BEHIND THE PAPERS

Today: Bruce Maggs

Bio: *Bruce Maggs is the Pelham Wilder Professor of Computer Science at Duke University. He received the S.B. (1985), S.M. (1986), and Ph.D. (1989) degrees in computer science from the Massachusetts Institute of Technology. His advisor was Charles Leiserson. After spending one year as a Postdoctoral Associate at MIT, he joined NEC Research Institute in Princeton. In 1994, he moved to Carnegie Mellon, where he stayed until joining Duke in 2009. While on a two-year leave-of-absence from Carnegie Mellon, Maggs helped to launch Akamai Technologies, leading all engineering efforts as its Vice President for Research and Development. In 2018 he was one of the inaugural winners of the SIGCOMM Networking Systems Award for the Akamai content delivery network, and was named a Fellow of the ACM. From 2019 to 2024 he served as Director of Engineering for Emerald Innovations.*



EATCS: We ask all interviewees to share a photo with us. Can you please tell us a little bit more about the photo you shared?

Bruce: I go windsurfing whenever I can! One weekend last winter I flew to Tampa because the wind forecast looked promising. Alas, the weather didn't cooperate. The wind was light and the air was cold (8°C). As you can see from the photo, I went out anyway.

EATCS: Can you please tell us something about you that probably most of the readers of your papers don't know?

Bruce: I had an unusual childhood for an American boy. My hometown is Urbana, Illinois. My father is a law professor and when I was a boy his speciality was Soviet law. When I was young we lived for months at a time in Bulgaria, Yugoslavia, Romania, and the Soviet Union. (We also lived in Hawaii for eight months.)

My father was also an early computer hobbyist. I remember watching him lying on his stomach in the living room filling out IBM coding sheets. Later he would go to the university and punch the corresponding cards and then run his programs. In 1976, when I was thirteen, he had a PLATO computer terminal installed in our home, connected by a 1200-baud dedicated line. PLATO was an innovative mainframe-based computer system developed jointly by Control Data and the University of Illinois. The creators of PLATO also invented the plasma panel. PLATO was intended to enable computer-aided instruction, but it really excelled at multi-player games. At first I was only interested in playing these games. My friend Andrew Shapira and I worshipped the game authors and eventually we decided that we wanted to be the objects of this adulation. So in junior high school we started writing a dungeons and dragons game called "Avatar." We were soon joined by a third author, C. David Sides and received some help from Mike Berger. Avatar eventually became the most popular program on the PLATO system.

EATCS: Is there a paper which influenced you particularly, and which you recommend other community members to read?

Bruce: As a graduate student I was blown away by Bob Tarjan's analysis of the union-find algorithm [5, 6], which, I think, was the first analysis to incorporate the inverse Ackermann function in the running time of an algorithm. It was hard for me to imagine where his "multiple partition" method had come from. The paper demonstrates that seemingly intractible problems can solved with enough ingenuity. But I can't limit myself to one paper! Others that had a big impact on my work were the papers that describe the AKS sorting network [1, 3], and the papers that introduce Valiant's idea of first routing to a random destination [7, 8].

EATCS: Is there a paper of your own you like to recommend the readers to study? What is the story behind this paper?

Bruce: I guess my most “famous” paper was co-authored with Tom Leighton and Satish Rao. Simplifying just a little, it shows that in any network, any packet routing problem can be solved in $O(c + d)$ steps, where c is the congestion of the paths of the packets in the network, and d is the dilation [2]. Congestion is the maximum number of packets that cross any edge of the network, and dilation is the length of the longest path. The key to the paper is a beautiful (if I may say so) application of the Lovász Local Lemma [4, pp. 57–58]. As a graduate student I presented this paper at FOCS. It was my first such presentation. Later I shared an elevator with Bob Tarjan (see above) who I hadn’t met yet. He said, “Nice talk.”

EATCS: When (or where) is your most productive working time (or place)?

Bruce: I’m often so busy during the day that I can’t think straight. Over and over again I’ve seen the solution to a problem just after lying down to sleep at night, when I can finally relax. Sometimes this makes me feel like I didn’t try very hard during the day!

EATCS: What do you do when you get stuck with a research problem? How do you deal with failures?

Bruce: I feel like I’m stuck 99% of the time. When I’m frustrated I remind myself that it’s possible to have a highly successful research career with only one or two breakthroughs per year. Then I go back to banging my head against the wall. But, as I was taught by Tom Leighton, one of my mentors in graduate school, after a breakthrough occurs you must explore the consequences with all your energy!

EATCS: Is there a nice anecdote from your career you would like to share with our readers?

Bruce: I transferred to MIT as a junior in college. The very first class I attended was 6.045, “Automata, Computability and Complexity Theory,” taught by Charles Leiserson. I loved this subject. At the end of the semester I approached Charles and asked if he would be willing to supervise an undergraduate research project on complexity theory. But he told me that complexity theory wasn’t really his research area. He said, however, that he would be happy to introduce me to some outstanding faculty members at MIT who worked in the area. I thought for just a moment, and then said no, I really liked the way he presented the material in class, and I wanted to work with him. So I asked what his research was about. Eventually he became my Ph.D. advisor. Choosing to work with Charles was the best career decision I ever made.

EATCS: Do you have any advice for young researchers? In what should they invest time, what should they avoid?

Bruce: Invest your time trying to solve problems that are important to you. Your motivation will be better, and your work will be more likely to have an impact. You might also consider taking some time off from academia to work at start-up companies. I really enjoy the teamwork aspect of building a new company. By necessity you'll learn many new things! At my first start-up, Akamai Technologies, I learned a lot about computer networks and distributed systems and decided to change my research focus from the theory of parallel algorithms and architectures to computer networking and distributed systems. My former Ph.D. student Anja Feldmann served as my mentor as I transitioned into the computer networking community. Most recently I've been working at a start-up called Emerald Innovations. Emerald is an MIT spin-off founded by Dina Katabi and three of her former Ph.D. students, Rumen Hristov, Hariharan Rahul, and Zac Kabelac. Emerald has developed a "human radar" device that can track the breathing, sleep cycles, and mobility of a person at home. The Emerald device collects data in the homes of volunteer participants in clinical drug trials to help assess the effectiveness of new medications.

EATCS: What are the most important features you look for when searching for graduate students?

Bruce: By far the most important trait is persistence. The most successful graduate students just don't give up. When one approach fails, they try another. Each time they come to your office they surprise you with something they found under a new rock they turned over.

EATCS: Do you see a main challenge or opportunity for theoretical computer scientists for the near future?

Bruce: The biggest challenge remains translating a real-world problem into a theoretical framework in a way that's amenable to analysis, but still provides guidance on solving the original problem. I don't know that I'm particular adept at that.

Please complete the following sentences?

- *Being a researcher is a privilege! Very few people get to decide what is most important to solve and go after it.*
- *My first research discovery was a “communication efficient” algorithm for computing minimum-cost spanning trees on a fat-tree parallel computer.*
- *Enjoying research is the key to being a happy academic. It makes up for all the overhead that academic life entails.*
- *Theoretical computer scientists 100 years from now will look back and marvel that primitive humans struggled to prove $P \neq NP$.*

References

- [1] M. Ajtai, J. Komlós, and E. Szemerédi. Sorting in $c \log n$ parallel steps. *Combinatorica*, 3:1–19, 1983.
- [2] F. T. Leighton, B. M. Maggs, and S. B. Rao. Packet routing and job-shop scheduling in $O(\text{congestion} + \text{dilation})$ steps. *Combinatorica*, 14(2):167–180, 1994.
- [3] M. S. Paterson. Improved sorting networks with $O(\log N)$ depth. *Algorithmica*, 5:75–92, 1990.
- [4] J. Spencer. *Ten Lectures on the Probabilistic Method*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1987.
- [5] R. E. Tarjan. Efficiency of a good but not linear set union algorithm. *Journal of the ACM*, 22(2):215–225, Apr. 1975.
- [6] R. E. Tarjan. *Data Structures and Network Algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1983.
- [7] L. G. Valiant. A scheme for fast parallel communication. *SIAM Journal on Computing*, 11(2):350–361, May 1982.
- [8] L. G. Valiant and G. J. Brebner. Universal schemes for parallel communication. In *Conference Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computing*, pages 263–277, May 1981.

The Bulletin of the EATCS

The Bulletin of the EATCS

THE VIEWPOINT COLUMN

BY

STEFAN SCHMID

TU Berlin, Germany
stefan.schmid@tu-berlin.de

THE ROLE OF AN INVITED SPEAKER AT A CONFERENCE

Luca Aceto
ICE-TCS, Department of Computer Science,
Reykjavik University
Gran Sasso Science Institute, L'Aquila
luca@ru.is, luca.aceto@gssi.it

I recently attended a meeting during which the topic of invited speakers at conferences came up. As part of the ensuing discussion, a colleague mentioned that, in an ideal world, an invited speaker should “engage with the conference”, as typically indicated in the message of invitation they receive from the conference organisers. However, in the opinion of that colleague, experience shows that it is increasingly common for invited speakers to show up at the conference to deliver their talk, stay for a few hours and then fly back home. Another colleague stated that they would rather have more contributed talks and fewer invited talks at conferences, if any at all, as they prefer to hear about the latest developments in their specific research area rather than listen to a longish talk devoted to a topic that is not directly connected to their main scientific interests. On a similar note, I recall meeting a colleague from “Volume A” theoretical computer science on my way to attending Xavier Leroy’s¹ invited talk at ICALP 2016² and asking him whether he was coming too. His reply was “No. That would be like going to a talk in the life sciences!”, or something in that spirit. I suspect that several “Volume B” colleagues would have given me a similar answer if I encouraged them to attend a “Volume A” keynote address.

These are all valid observations and points. However, in the humble and probably old-fashioned opinion I aired during that meeting, invited talks should be one of the occasions in which the whole research community of reference for a conference comes together, learns something from some of its leading or up-and-coming scientists who are also excellent communicators, showcases role models to its younger scientists, celebrates the unity of theoretical computer science (broadly construed) and possibly sows the seeds for serendipitous cross-fertilisation of

¹See <https://xavierleroy.org/> for his web page.

²The slides are available at <https://xavierleroy.org/talks/ICALP2016.pdf>.

ideas between its sub-fields.

So, assuming anyone shares my Jurassic viewpoint on the role of invited speakers at our conferences, how should a “model invited speaker” behave to repay the investment that their research community made in inviting them, to justify the trust placed in them and to do themselves and their work justice? To answer this question and to express my views in a hopefully convincing way without being too boring, let me instead describe the behaviour of an invited speaker, whom I will call Prof. X (no relationship with Prof. Charles Francis Xavier³), that I would consider to be the antithesis of a model keynote speaker at a conference and possibly a nightmare for the conference organisers and participants.

Before the conference Having received an invitation to deliver an invited talk at a prestigious conference from its organisers, Prof. X looks at their schedule and realises that they might have time to attend only one conference day. Despite their lack of time, Prof. X accepts the invitation, but does not inform the organisers that their participation will be limited to at most one conference day. Close to the conference dates, when asked about their travel plans, Prof. X finally tells the organisers that they will arrive the evening before their talk is scheduled and that, due to other engagements that cannot be moved, they will have to leave the conference after having delivered the talk. Unfortunately, the travel costs are high and Prof. X does not have a research grant that can cover the cost of airfare.

At the conference Prof. X arrives at the conference venue at 8:55 AM, five minutes before their talk is scheduled, rushes to the dais and connects their laptop to the projector. Fortunately, there are no technical problems and the organisers and the session chair sigh in relief. Prof. X delivers their invited talk, making sure that the talk has no clear, take-home message, carefully eschewing the context for, and the high-level ideas behind, the research covered in the talk and focusing solely on the most technical aspects of the work. Apart from a handful of conference participants who are deeply familiar with Prof. X’s recent work, the majority of the audience is lost already after the title slide.

Despite several signals from the session chair, Prof. X’s talk runs ten minutes overtime, leaving no time for questions from the audience and shortening the subsequent coffee break. While the audience streams out of the lecture room and heads for what is left of the coffee break, two PhD students approach Prof. X since they’d like to ask some questions related to the presentation that are relevant for their doctoral research. Unfortunately, Prof. X has no time for them. He has booked a taxi to the airport and the driver is already waiting.

³https://en.wikipedia.org/wiki/Professor_X

After the conference As soon as Prof. X is back in their office, they send a reimbursement claim to the conference organisers, including items that were not mentioned by the organisers in their letter of invitation. Prof. X also reads an email from the two PhD students who approached her/him after the talk. The email contains some clearly worded questions related to Prof. X's work and explains how those questions are connected to the students ongoing research for their doctoral studies. Alas, Prof. X never replies.

Of course, what I just wrote is a caricature and there are nuances and exceptions that I have swept under the carpet. Indeed, many of us need to balance family life and the ever-increasing demands of our academic work⁴, as well as attending conferences with our teaching obligations that may make it hard to reschedule lectures or to find a colleague who is willing to cover for us when we travel during term time. However, to my mind, being invited to deliver a keynote address at one of our conferences is a great honour and also a great responsibility. Next time we are asked to be an invited speaker, perhaps we should accept the invitation only if we can attend most of, ideally all, the conference and contribute to its success by attending talks, asking questions, discussing with colleagues during the coffee breaks and mentoring the junior members of the research community. Moreover, when we know that, mostly likely, we will only be able to be at the conference for a day or two, let's inform the organisers right away and ask them whether what we can offer is agreeable to them.

If we accept to deliver a talk, let's make sure that we give the audience something to take home⁵ and that we highlight the message we want to convey repeatedly, following the motto "tell them what you are going to tell them, tell them, and tell them what you've told them." Having a slide deck on some topic is not the same as having a message and an engaging story line to deliver that message to a broad audience. In fact, in my humble opinion, what I just wrote applies to every talk and most of us would benefit by paying heed to Gian-Carlo Rota's⁶ four requirements for a good talk:

1. Every talk should make only one main point.
2. Never run overtime.
3. Relate to your audience.
4. Give them something to take home.

⁴Conference organisers might consider offering some form of childcare. I feel that doing so would help to increase diversity amongst the conference attendees, by making it easier for colleagues to attend even when they have young children.

⁵See <https://www.ams.org/notices/199701/comm-rota.pdf>.

⁶See https://en.wikipedia.org/wiki/Gian-Carlo_Rota.

However, even a model invited speaker is nothing without an audience! To my mind, we should strive to attend every invited talk at the conferences we attend, even those in research fields that aren't exactly our own. Doing so will expand our horizons and nurture our intellectual curiosity at the small price of at most one hour of our time. Who knows, perhaps the message delivered by a model invited speaker will turn out to have some relevance for our research at some point in the future. Even if that never happens, we will have learnt something we did not know before attending that talk, which I believe is worthwhile in itself. Moreover, to my mind, we should give our students and junior colleagues a good example. If we do not attend (invited) talks ourselves, why should they?

As Rota wrote in his thought-provoking piece “Ten lessons I wish I had been taught”⁷,

“The advice we give others is the advice that we ourselves need.”

For the little that it may be worth, I will try to heed the aforementioned advice myself.

Acknowledgments I thank Aggela Chalki, Magnús M. Halldórsson, Eva Rotenberg, Stefan Schmid and Jana Wagemaker for reading versions of this text and for their feedback on it. Of course, I am solely responsible for the contents of this piece and for any infelicities in it.

⁷Available at <https://www.ams.org/notices/199701/comm-rota.pdf>.

TCS ON THE WEB

BY

STEFAN NEUMANN

TU Wien
Erzherzog-Johann-Platz 1
1040 Vienna, Austria
stefan.neumann@tuwien.ac.at
<https://neumannstefan.com>

Clément Canonne is a senior lecturer at the University of Sydney. He works on distribution and property testing, the theory of machine learning, and differential privacy. He recently received the 2024 Caspar Bowden Award for Outstanding Research in Privacy Enhancing Technologies (joint with Gautam Kamath and Thomas Steinke) and was a co-organizer of the Sublinear Algorithms program at the Simons Institute in Berkeley, California.

Clément is also one of the most important theoretical computer science influencers on Twitter/X with more than 30,000 followers.¹ In his guest column, he answers our questions on how he started tweeting (or “microblogging” as people used to call it) about theoretical computer science (TCS), and how we can use social media to promote TCS to a broader audience.

¹https://twitter.com/ccanonne_

PROMOTING THEORETICAL COMPUTER SCIENCE ON SOCIAL MEDIA

A Conversation with Clément Canonne

Clément, thank you for taking time for this interview. You do not have a blog, but you are one of the most-influential theoretical computer scientists (TCS) on Twitter/X with more than 30,000 followers.¹ So, I thought featuring you in this column would be a good idea. You joined Twitter in 2011, do you still remember what motivated you to go there? What has changed since then?

I think I joined Twitter on a whim, not being very clear what that was good for – then several years passed, during which I forgot about my account. I became more active around 2019 or so, as it seemed some TCS researchers were posting there; and then, the pandemic happened, and I started spending a significant amount of time on Twitter. It was a way to stay connected with other researchers, and what was happening, and to have at least some meaningful TCS-related interactions and discussions beyond my living room at the time.

Now, a lot has changed, for clear reasons: first, the change of ownership of the platform, which has affected the form and content in many ways. Another more personal aspect is my move to Australia, which has meant a slightly different experience: now including people from a different continent, but also a change of rhythm and content I see and gets brought to me by Twitter, due to the time zone. It's an interesting shift.

Is there a reason why you share your thoughts on Twitter/X and not in a more classic blog format?

Good question! I really enjoy some of the TCS blogs (e.g., Boaz Barak's "Windows On Theory", to name one among others), but I think I lack the discipline to write long form in a regular fashion. I also think Twitter has (or used to have?) the appeal of conciseness, which is a valuable exercise in itself: how to convey a point or idea in very few characters. I'm not claiming all the ideas or points I'm conveying are good, or worth writing, though... I don't think I'd have

¹https://twitter.com/ccanonne_

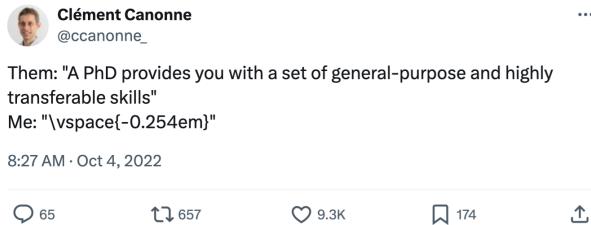


Figure 1: One of Clément’s viral tweets.

much credibility there, given that my Twitter account is roughly 50% puns, 50% technical content, and 50% dubious math.

For what it’s worth, I tend to like long form writing in a different context, that of writing lecture notes or surveys; and I am an occasional writer on blogs such as <https://differentialprivacy.org>.

Are there some posts or interactions that you would like to highlight?

Not necessarily the most informative, but I remember once waking up and making a silly CS-related pun. I think it was about LaTeX, something about negative \vspace (see Figure 1, the editor). It went sort of viral, I guess LaTeX woes are universal in our field: three weeks later, I received a message from a friend from undergrad in France about it, telling me my name popped up in a conversation with a colleague. Not sure what to make of that, except that I didn’t have “being talked about in my home country because of a LaTeX tweet” on my bingo card.

Maybe more informative though is the fact that having some reach (moderate, I won’t kid myself, but still) can be useful – for instance, I want to believe that repeatedly highlighting the Australian student visa issues on Twitter (back in 2021–2022) did make a difference. It definitely ended up gathering some amount of media attention, and (hopefully?) helped a little.

A lot of theory researchers mostly use social networks to promote their own research and new results in their own area. However, your content has a much broader scope, ranging from classic textbook algorithms, over probability puzzles to new results in your area. Do you have a specific process of what you post about?

I wish I had! Honestly, I feel like most of my Twitter posts come from various things I read, or hear about (talks, lecture notes, textbooks, interesting nuggets in papers) and want to share. There is no real process beyond the common thought

that “if I had known that 10 years ago, I’d have liked it.” I have a tendency to like neat, self-contained “tricks” or results, and my working assumption is that others will want to learn about them too. (There’s also a whole other part of my posts that comes from jokes or puns which pop up, also often reading papers or textbooks. No real process there either...)

Has your active presence on Twitter/X helped you to find new research ideas or even new co-authors?

It has definitely made me think more deeply about some questions: for instance, after posting something and seeing the comments and discussions, realizing that one assumption in the phrasing was not obvious, or at least less natural than I thought. I guess it’s part of a broader picture related to science communication in general, where we don’t fully see our own thinking patterns and biases until we have to explain an idea to a broad audience.

In terms of research ideas, the main advantage of Twitter has been the exposure to other people’s research. It’s hard to keep track of all the interesting stuff that happens, but people actually explaining their new paper in a few tweets (which is hard!) is a very good way to at least get some of it. Recently I ended up using a result from a paper I’m confident I’d have completely missed had it not been discussed on Twitter: the title would not have rung a bell, the abstract wouldn’t have seemed relevant, and I probably would have never read it.

Sometimes my impression is that in TCS we have a lot of great results, that would deserve more visibility in the general CS community. Do you have some suggestions to other TCS researchers on how we can improve our outreach on social media?

That’s quite tricky, though related to the previous answer. I don’t really have a good idea beyond stuff that worked for me somehow, and things I enjoy from others. Generally speaking, I enjoy to learn things, even if they might seem simple: one “cool trick” from a paper goes a longer way than a very technical and overwhelming statement. And it’s nice to spend time mentioning and discussing such great results even when they’re not yours (especially when they’re not): not to say there cannot be any self-promotion on social media, of course! But at the end of the day, the goal is to make as many people as possible aware of all the incredible stuff there is, regardless of whom it came from.

Before we conclude the interview, could you please let us know what your current research is about?

One thing that has been on my mind for a few years now is what happens to a lot of classical statistical or computational learning questions (including “well understood” ones) when you see everything as a “resource”. The first resource we learn about in CS undergrad is time, and that’s the first foray into algorithm analysis. But memory is a constraint as well, and communication in distributed settings, and randomness, too. And once you’ve started, you realize you can view privacy as another, and robustness to adversarial corruptions, and maybe even the type of “queries” that are allowed, and how much adaptivity your algorithms can have. Now you have a range of constraints and resources, and analyzing their interplay and the various tradeoffs is fascinating, mathematically and algorithmically. There’s so much of it we don’t understand, even for the most basic tasks!

Finally, is there anything else you want our readers to know?

Wombats are really cute animals. Seriously, they’re like little fluffy marsupial juggernauts.

Oh, also, and on a more serious note: if you are not already, please check out the TCS Blog Aggregator at <https://theory.report/> to stay informed about TCS-related papers, blog posts, announcements, and job postings. I’m not affiliated with it, it’s just a great resource.

Thank you for this nice interview!

THE DISTRIBUTED COMPUTING COLUMN

Seth Gilbert

National University of Singapore

seth.gilbert@comp.nus.edu.sg

This month, the Distributed Computing Column is featuring Robin Vacus, winner of the 2024 Principles of Distributed Computing Doctoral Dissertation Award. His thesis on “Algorithmic Perspectives to Collective Natural Phenomena” explores how distributed computing techniques can help to understand biological processes found in the real world. It examines a variety of different phenomena, including problems in synchronization (i.e., agreement), problems in task selection, and problems in cooperative behavior. Quoting from the award citation: “Dr. Vacus’ thesis provides an inspiring overview of the questions studied, and employs a wide range of tools and techniques, involving probabilistic analysis, control theory, statistics and game theory, and computer simulations.”

This column focuses on the question of “bit dissemination,” a basic agreement problem in which a single source wants to disseminate one bit information to all the agents in the system. Specifically, it discusses several different self-stabilizing protocols for achieving dissemination based only on passive communication, and it explores issues of limited memory and limited communication (i.e., sample size).

The Distributed Computing Column is particularly interested in contributions that propose interesting new directions and summarize important open problems in areas of interest. If you would like to write such a column, please contact me.

MINIMAL REQUIREMENTS FOR BIT-DISSEMINATION WITH PASSIVE COMMUNICATION

Robin Vacus

Multi-agent systems often face two closely related challenges: achieving consensus among agents (i.e., making sure they take consistent decisions), and aggregating local information about their environment. Indeed, reaching approximate agreement is often a prerequisite of collective decision-making, while the overall performance of the group depends on how effectively local information is processed. For example, consider a group of ants carrying a heavy load. In the first place, in order to prevent forces from canceling each-other and successfully move the object, almost all ants in the group must push in the same direction – which requires a degree of consensus. In addition, information about an optimal route to the nest, which may be distributed unevenly among the ants, must be processed in order to enable the group to reach its destination quickly [11, 8]. These tasks become especially difficult when the system is susceptible to faults or when interactions between agents are noisy and constrained. This is particularly evident in biological ensembles, where individuals may have a wide range of internal computational abilities, but often limited communication capacity. Addressing these challenges and limitations in a biologically relevant way necessitates specific modeling approaches, which is why the bit-dissemination problem was introduced [4].

1 The Bit-Dissemination Problem

Problem definition. We consider a discrete time process over a fully-connected network of n agents. In round t , each agent i holds an *opinion* $X_t^{(i)} \in \{0, 1\}$. In each execution, one of these opinions is called *correct* and denoted z . The group contains one *source agent* i^* which knows which opinion is correct, and must always remain with it: for every t , $X_t^{(i^*)} = z$.

We adopt the basic \mathcal{PULL} model of communication: each time a non-source agent is *activated*, it obtains a random vector $S \in \{0, 1\}^\ell$, in which every element is equal to 1 with probability $x_t := \frac{1}{n} \sum_i X_t^{(i)}$ (the proportion of agents with opinion 1 in that round) and 0 otherwise. This is equivalent to sampling the opinions of ℓ other agents taken uniformly at random. Typically, the *sample size* ℓ shall

be negligible compared to the size of the group n . After receiving S and depending on the protocol, the activated agent may decide to update its opinion $X_t^{(i)}$ and memory state. We distinguish between two different activation settings: the *parallel* setting, where all agents are activated simultaneously in every round, and the *sequential* setting, where only one non-source agent selected uniformly at random is activated.

Relying only on the passive presence of the source, the goal of non-source agents is to reach a consensus on the correct opinion as fast as possible, and remain with it forever. Moreover, we restrict attention to *self-stabilizing* protocols, able to converge fast regardless of the initial configuration of the system. Specifically, a protocol is considered correct if, starting from any initial distribution of opinions (including the correct opinion z) and any initial memory states of the agents, it reaches a configuration where all $X_t^{(i)}$ are equal to z within poly-logarithmic time, with high probability.

Key quantities. Our objective is to pinpoint the minimal requirements for solving the bit-dissemination problem. We focus on the two following resources:

- The memory of the agents, i.e., the amount of information about past observations (measured in bits) that they are allowed to use in order to make a decision.
- The sample size ℓ . Intuitively, the larger ℓ is, the more accurately the agents can estimate x_t (the proportion of 1-opinions in the population).

For convenience, we express the convergence time of protocols in both activation settings using *parallel rounds*. One parallel round corresponds to n activations, which equals 1 round in the parallel setting and n rounds in the sequential setting. Keep in mind that while this allows for qualitative comparisons, the two settings are not directly equivalent.

Hardness induced by self-stabilization. By definition of self-stabilization, protocols cannot specify the initialization of the memory of the agents, which should be thought of as being set up by an adversary. As a consequence, agents cannot rely on having a shared clock modulo- T from the beginning of execution, or even distinct identifiers. Instead, a protocol must be able to create and maintain such objects from any arbitrary state in order to use them.

Hardness induced by passive communications. The fact that sampling happens over the opinions of other agents is often referred to as *passive communication*. This assumption is inspired by natural scenario where information can be

gained only by observing the behavior of other agents, which, in principle, may not even intend to communicate [5]. Not only does it restrict each observation to a single bit in our model, but it also leaves the agents essentially unable to communicate when it comes to reaching consensus, since they are then forced to display the correct opinion z .

In contrast, a distributed system is said to exhibit *active communications* if the opinion $X_i^{(t)}$ of each agent i differs from the 1-bit message displayed to other agents. Consensus would then be defined over the opinion $X_i^{(t)}$, while sampling would occur separately, based on the arbitrary 1-bit message. Results from [1], allowing agents to synchronize a clock in a self-stabilizing way, can be used to solve the bit-dissemination problem with active communications. Moreover, a solution to an equivalent problem (called “bit-broadcast”) was identified for population protocols [7]. However, neither of these algorithms can be adapted to the framework of passive communications, which appears to be significantly weaker. These considerations are discussed in more detail in [9, Section 1.4].

2 Fast Dissemination with Memory

In this section, we investigate what happens when agents are allowed to use a moderate amount of memory. We describe a simple algorithm, called *Follow the Trend (FtT)*, that efficiently solves the bit-dissemination problem while being self-stabilizing.

The protocol is based on letting agents estimate the current tendency direction of the dynamics, and then adapt to the emerging trend. Informally, each non-source agent compares the number of 1-opinions that it observes in the current round, with the number observed in the previous round. If more 1’s are observed now, then the agent adopts the opinion 1, and similarly, if more 0’s are observed now, then it adopts the opinion 0. If the same number of 1’s is observed in both rounds, then the agent does not change its opinion. Formally, our algorithm is defined as [Algorithm 1](#) (the number of 1-opinions observed in the last round is stored in a variable named σ_t).

As illustrated on [Figure 1](#), this behaviour creates a persistent movement of the average opinion of non-source agents towards either 0 or 1, which “bounces” back when hitting the wrong opinion.

Up to modifying it slightly (to account for technical difficulties), we can show that [Algorithm 1](#) solves the bit-dissemination problem efficiently when all agents are activated simultaneously, as long as the sample size ℓ is at least poly-logarithmic in n .

Theorem 1 (Theorem 1 in [9]). *There exists a protocol based on [Algorithm 1](#) that solves the bit-dissemination problem in the parallel setting in $O(\log^{5/2} n)$*

Algorithm 1: Follow the Trend (sketch)

```

1 Input: Current opinion  $X_t^{(i)} \in \{0, 1\}$ , memory state  $\sigma_t \in \{0, \dots, \ell\}$ , opinion sample  $S \in \{0, 1\}^\ell$ 
2  $\sigma_{t+1} \leftarrow$  number of 1-opinions in  $S$  ;
3 if  $\sigma_{t+1} > \sigma_t$  then  $X_{t+1}^{(i)} \leftarrow 1$  ;
4 else if  $\sigma_{t+1} < \sigma_t$  then  $X_{t+1}^{(i)} \leftarrow 0$  ;
5 else  $X_{t+1}^{(i)} \leftarrow X_t^{(i)}$  ;

```

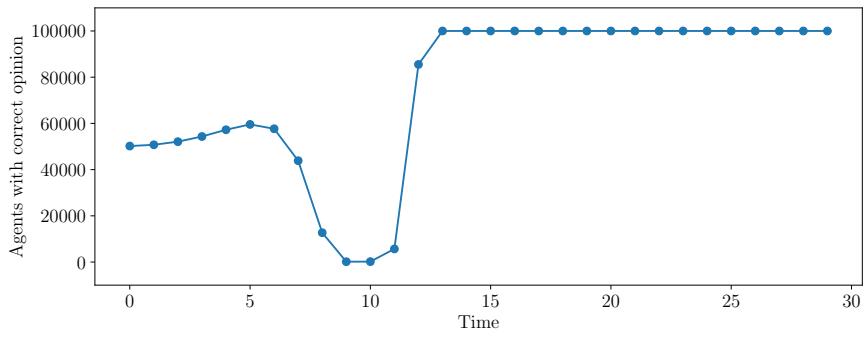


Figure 1: One execution of [Algorithm 1](#) in the parallel setting, with $n = 10^5$ and $\ell = 34$. The graph shows the number of agents holding the correct opinion as a function of the round number.

parallel rounds with high probability, while relying on $\ell = \Theta(\log n)$ samples and $\Theta(\log \log n)$ bits of memory.

To prove the correctness of a self-stabilizing algorithm, there are generally 2 methods. Ideally, one is able to exhibit a scalar quantity (a “potential”) that contains enough relevant information about the system, while being simpler to analyze. For example, if this quantity is simultaneously decreasing at a fast rate and bounded from below, then it directly implies an upper bound on the convergence time of the algorithm. Another, more tedious approach is to partition the configuration space into as many subsets as necessary, and then characterize the algorithm’s behavior on each of the subsets. Unfortunately, our proof of [Theorem 1](#) belongs to the latter category. When [Algorithm 1](#) is used, each configuration is fully characterized by the proportion of 1-opinions in the previous round and in the current round, i.e., by the couple (x_t, x_{t+1}) . The resulting 2-dimensional configuration space is depicted in [Figure 2](#), together with the partition used in the proof of [Theorem 1](#).

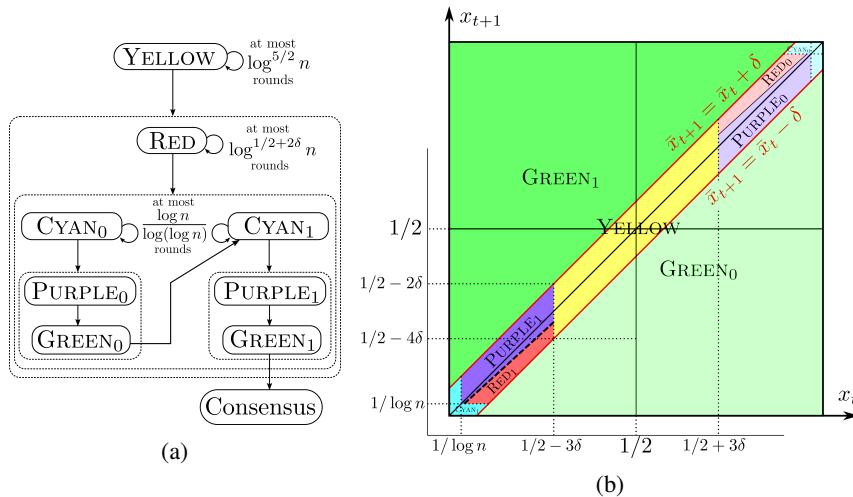


Figure 2: **(a)** Sketch of the proof of Theorem 1. Self-loops indicate the number of rounds spent by the process in corresponding areas. **(b)** An illustration of the areas (see [9, Section 2.1] for an exact definition).

Although Theorem 1 only holds in the parallel setting, it seems that the ‘‘Follow the Trend’’ strategy can be adapted to the sequential setting (where only one random agent is activated at a time). The trick is to let agents execute one step of Algorithm 1 every $\log n$ activations. This technique essentially limits the number of time that an agent i can be executed between two activations of another

agent j , at the cost of slowing down the process by a logarithmic factor. While the resulting protocol was never rigorously analyzed, a detailed description and some empirical evidence about its correctness can be found in [2, Section 5].

3 The Need for Memory in the Sequential Setting

In this section, we restrict attention to *memory-less* algorithms, in order to further characterize the requirements of the problem. More specifically, we consider update rules that only depend on the agent’s current opinion $X_t^{(i)}$, and on the last sample S . In particular, this assumption precludes the possibility to maintain clocks and counters, or to estimate the tendency of the dynamics as in [Algorithm 1](#).

Since agents having the same opinion are indistinguishable in absence of memory, the configuration of the system in round t is fully described by the proportion x_t of agents with opinion 1. Moreover, the distribution of x_{t+1} depends only on x_t , and hence the process is always a Markov chain on $\{0, \frac{1}{n}, \dots, \frac{n-1}{n}, 1\}$. In the sequential setting, the number of agents with opinion 1 may only vary by at most one unit in every round. The Markov chain is even simpler in this case, as its underlying graph is just a path of length $n + 1$. These are known as “birth-death” chains [10, Section 2.5]. The exact transition probabilities of the birth-death chain induced by a given protocol \mathcal{P} does not only depend on \mathcal{P} , but also on the opinion of the source (which is not running the protocol). For example, the state $x_t = 0$, corresponding to a consensus on opinion 0, cannot be reached when the source has opinion 1. However, since the source itself is sampled with probability only $1/n$ by other agents, its impact on most of the transition probabilities is expected to be quite limited.

In summary, a protocol \mathcal{P} solving the bit-dissemination problem in the sequential setting implies the existence of two nearly identical birth-death chains C_1 and C_0 (corresponding to the case that the correct opinion is 1 or 0 respectively). Due to their similarity, any tendency of C_1 to move towards state 1 (a consensus on opinion 1) implies an almost equivalent tendency of C_0 to move away from state 0 (a consensus on opinion 0). This observation can be leveraged to obtain the following lower-bound, which interestingly, holds regardless of the sample size.

Theorem 2 (Theorem 3 in [2]). *The expected convergence time of any memory-less dynamics for the bit-dissemination problem in the sequential setting is at least $\Omega(n)$ parallel rounds, or equivalently $\Omega(n^2)$ activations, even when $\ell \geq n$.*

The same reasoning implies that the lower bound is reached when the induced birth-death chains are unbiased random walks. The corresponding algorithm turns out to be the *voter* dynamics ([Algorithm 2](#)), in which activated agents simply copy

the opinion of another agent chosen uniformly at random. Indeed, an upper bound based on [Algorithm 2](#) and almost matching [Theorem 2](#) follows from classical arguments.

Algorithm 2: Voter dynamics ($\ell = 1$)

- | |
|---|
| 1 Input: Opinion sample $S \in \{0, 1\}$ |
| 2 $X_{t+1}^{(i)} \leftarrow S;$ |

Theorem 3 (Theorem 4 in [2]). *The voter dynamics solves the bit-dissemination problem in the sequential setting in $O(n \log n)$ parallel rounds in expectation.*

Since the voter dynamics only uses $\ell = 1$, we conclude that the sample size is of no importance in the sequential setting. On the other hand, together with the insights from [Section 2](#), our results imply that memory is a critical requirement.

4 The Minority Dynamics

In contrast to the sequential setting, memory-less dynamics in the parallel setting may jump from any configuration to any other in just one round – albeit with extremely small probability. Therefore, the ideas behind [Theorem 2](#) are not applicable, which suggests that the lower bound could be beaten. In this section, we show that fast convergence is indeed possible, by considering the *minority* dynamics ([Algorithm 3](#)). Introduced by Amos Korman, this fascinating protocol has its own interest beyond the bit-dissemination problem.

The minority rule is defined as follows: if an agent sees unanimity among the ℓ elements of the sample S , it adopts this unanimous opinion; otherwise, it adopts the opinion corresponding to fewer samples.

Algorithm 3: Minority dynamics

- | |
|---|
| 1 Input: Opinion sample $S \in \{0, 1\}^\ell$ |
| 2 if all opinions in S are equal to x then |
| 3 $X_{t+1}^{(i)} \leftarrow x;$ |
| 4 else |
| 5 $X_{t+1}^{(i)} \leftarrow$ minority opinion in S (breaking ties randomly); |

When all non-source agents follow [Algorithm 3](#), the proportion of agents with opinion 1 exhibits chaotic oscillations with no apparent pattern, as illustrated on

Figure 3. As long as the sample size is large enough, the oscillations eventually stop abruptly and the group reaches consensus in just a few rounds. However, when ℓ is too small compared to n , simulations depict seemingly endless oscillations. Identifying the value of ℓ required for the rapid convergence of the dynamics is a challenging task, as the chaotic nature of the process complicates the analysis. To this day, the only existing upper bound on the convergence time relies on a relatively large sample size ($\ell \geq \sqrt{n}$).

Theorem 4 (Theorem 1.3 in [3]). *If $\ell = \Omega(\sqrt{n \log n})$, then the minority dynamics solves the bit-dissemination problem in the parallel setting in $O(\log n)$ parallel rounds in expectation.*

Under the minority dynamics, the configuration of the system in round t is fully characterized by the number m_t of agents holding the minority opinion, and hence the configuration space is simply $\{0, \dots, \lfloor n/2 \rfloor\}$. As for [Theorem 1](#), the proof of [Theorem 4](#) proceeds by partitioning this one-dimensional space into several areas, in which the dynamics' behavior is easier to predict. It then consists in bounding the probability that the process remains stuck in the same area for a long time, and in showing that it eventually reaches the “green” area leading to consensus with constant probability (see [Figure 4](#) for an illustration).

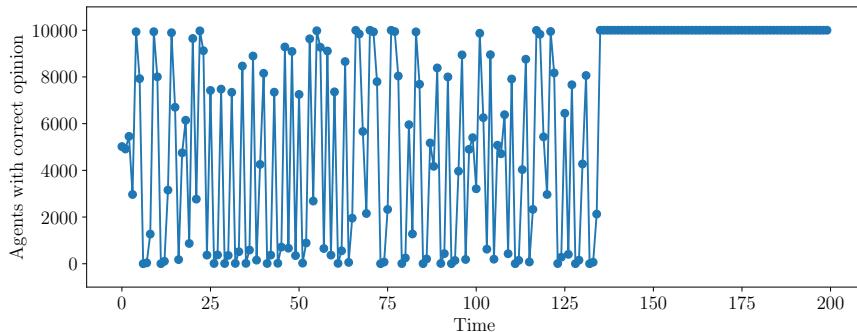


Figure 3: One execution of the minority dynamics in the parallel setting, with $n = 10^4$ and $\ell = 36$. The graph shows the number of agents holding the correct opinion as a function of the round number.

5 Open Questions

We conclude by listing the most interesting open problems arising from this line of works. We leave aside natural generalizations of the results described above,

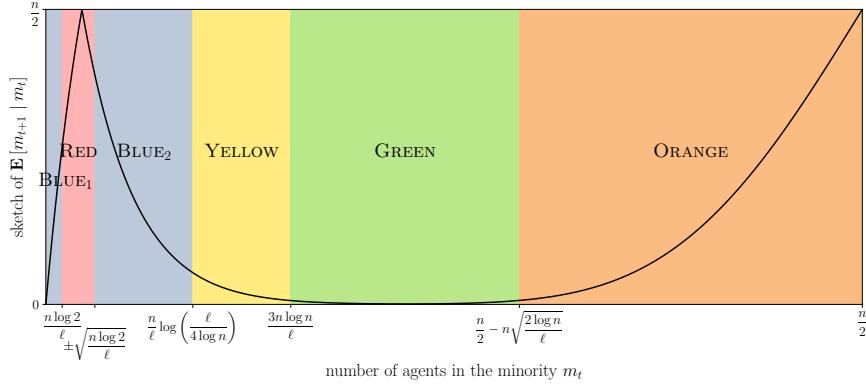


Figure 4: Partition of the configuration space into 6 areas used in the proof of [Theorem 4](#). The black line indicates how the random variable m_t , corresponding to the number of agents with the minority opinion, is expected to vary in the next round.

such as the case where agents have access to more than 2 opinions, or the case where the communication network is not fully connected.

- Q1)** *What is the smallest sample size ℓ^* allowing the minority dynamics to converge fast in the parallel setting?*

This question is interesting even in the absence of a source agent. It is possible to show that the convergence time of the minority dynamics is $\Omega(e^n)$ when $\ell = O(1)$, which together with [Theorem 4](#), implies that ℓ^* must satisfy $1 \ll \ell^* \leq \sqrt{n \log n}$ – leaving a huge gap for future works.

- Q2)** *Is there a memory-less algorithm solving the bit-dissemination with fewer samples than the minority dynamics in the parallel setting?*

As a first step towards answering this question, a general lower bound is given in [\[6\]](#). More specifically, it is shown that when $\ell = O(1)$, any memory-less dynamics needs almost linear time to solve the bit-dissemination problem, that is, at least $\Omega(n^{1-\varepsilon})$ rounds for every $\varepsilon > 0$. However, the arguments therein are not applicable even when ℓ is only $\Omega(\log n)$.

- Q3)** *Is there an algorithm solving the bit-dissemination problem when $\ell = O(1)$?*

Because of the aforementioned lower bound in [\[6\]](#), such algorithm would necessarily rely on memory. A promising candidate, called “BSF”, is mentioned in [\[12, Chapter 10\]](#), but was never successfully analyzed.

Acknowledgments

The results presented in this column were obtained as part of the doctoral thesis “Algorithmic Perspectives to Collective Natural Phenomena”, conducted under the supervision of Amos Korman and Pierre Fraigniaud.

References

- [1] Paul Bastide, George Giakkoupis, and Hayk Saribekyan. Self-stabilizing clock synchronization with 1-bit messages. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, Proceedings, pages 2154–2173. Society for Industrial and Applied Mathematics, January 2021.
- [2] Luca Becchetti, Andrea Clementi, Amos Korman, Francesco Pasquale, Luca Trevisan, and Robin Vacus. On the role of memory in robust opinion dynamics. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, {IJCAI-23}*, volume 1, pages 29–37, Macao, August 2023. International Joint Conferences on Artificial Intelligence Organization.
- [3] Luca Becchetti, Andrea Clementi, Francesco Pasquale, Luca Trevisan, Robin Vacus, and Isabella Ziccardi. The minority dynamics and the power of synchronicity. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, Proceedings, pages 4155–4176. Society for Industrial and Applied Mathematics, January 2024.
- [4] Lucas Boczkowski, Amos Korman, and Emanuele Natale. Minimizing message size in stochastic communication patterns: Fast self-stabilizing protocols with 3 bits. In *Proceedings of the 2017 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, Proceedings, pages 2540–2559. Society for Industrial and Applied Mathematics, January 2017.
- [5] Étienne Danchin, Luc-Alain Giraldeau, Thomas J. Valone, and Richard H. Wagner. Public information: From nosy neighbors to cultural evolution. *Science*, 305(5683):487–491, July 2004.
- [6] Niccolò D’Archivio and Robin Vacus. On the limits of information spread by memory-less agents. In *38th International Symposium on Distributed Computing (DISC 2024)*, Madrid, October 2024. Schloss-Dagstuhl - Leibniz Zentrum für Informatik.
- [7] Bartłomiej Dudek and Adrian Kosowski. Universal protocols for information dissemination using emergent signals. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2018, pages 87–99, New York, NY, USA, June 2018. Association for Computing Machinery.
- [8] Aviram Gelblum, Itai Pinkoviezky, Ehud Fonio, Abhijit Ghosh, Nir Gov, and Ofer Feinerman. Ant groups optimally amplify the effect of transiently informed individuals. *Nature Communications*, 6(1):7729, July 2015.

- [9] Amos Korman and Robin Vacus. Early adapting to trends: Self-stabilizing information spread using passive communication. *Distributed Computing*, February 2024.
- [10] David A Levin and Yuval Peres. *Markov Chains and Mixing Times, Second Edition*, volume 107. American Mathematical Soc., 2017.
- [11] H. F. McCreery and M. D. Breed. Cooperative transport in ants: A review of proximate mechanisms. *Insectes Sociaux*, 61(2):99–110, May 2014.
- [12] Emanuele Natale. *On the Computational Power of Simple Dynamics*. PhD thesis, 2016.

The Bulletin of the EATCS

The Bulletin of the EATCS

THE COMPUTATIONAL COMPLEXITY COLUMN

BY

MICHAL KOUCKÝ

Computer Science Institute, Charles University
Malostranské nám. 25, 118 00 Praha 1, Czech Republic

koucky@iuuk.mff.cuni.cz

<https://iuuk.mff.cuni.cz/~koucky/>

SEVEN WAYS OF DECIDING WHETHER MOST VERTEX SETS COVER A GRAPH

Till Tantau

Institute for Theoretical Computer Science

Universität zu Lübeck, Lübeck, Germany

till.tantau@uni-luebeck.de

Abstract

Given an undirected graph $G = (V, E)$, it is a difficult problem (complete for #P) to determine the total number of vertex covers $C \subseteq V$ of G . In contrast, deciding whether *most* C cover a graph (meaning, at least half of all possible $C \subseteq V$ are covers) turns out to be tractable. Intriguingly, this can be proved in very different ways, namely using search trees from FPT theory, using backdoor sets from SAT solving, using randomized sampling in conjunction with gaps in probability spectra, using algorithmic meta-theorems, using forbidden minors from graph minor theory, using forbidden subgraphs in conjunction with bounded tree-depth, and using the construction of algorithmically simple well-quasi-orderings. The present paper explains and celebrates how this diverse crowd of approaches, ideas, and methods, which Theoretical Computer Science has developed over the last fifty years, can be applied to solve a basic problem like “decide whether most vertex sets cover a given graph,” but also to solve many (at least superficially) harder counting-threshold problems.

1 Introduction

One of the central endeavours of Theoretical Computer Science in general, and of Algorithmics and Complexity Theory in particular, is to find different and ever better ways of solving computational problems. For some fundamental problems, such as matrix multiplication or sorting, there are impressive and seemingly endless (and not-yet-dried-up) streams of results in the literature that offer incremental runtime improvements in theory or in practice or both. Besides improving algorithms, one can also analyze problems from different viewpoints: For instance, the vertex cover problem and the clique problem are both NP-complete via extremely weak

reductions (and thus, in a sense, just “different encodings of the same question”), but these problems differ strongly when viewed from other viewpoints: The vertex cover problem is easily 2-approximated (just take all vertices in a maximal matching) while the clique problem does not even permit an $n^{1-\epsilon}$ -approximation (unless $P = NP$, see [35]), the vertex cover problem is fixed-parameter tractable [10], while the clique problem is W[1]-hard [9]. Studying computational problems using the different analytic tools developed by computer scientists over last fifty years, rather than focusing on a single algorithm for a problem, resembles exploring a landscape by hiking to different viewpoints and taking in the different views, rather than looking at a pixelated photo.

The landscape that we will explore in the present paper is centered on the following problem: “Given an undirected graph $G = (V, E)$, is it true that at least half of all possible vertex subsets $C \subseteq V$ are *covers* of G , meaning that each edge $\{u, v\} \in E$ intersects C ? ” For example, triangles  (also known as K_3) have this property: The empty set and the three one-element sets , , and  are no covers of the triangle, while the three two-element subsets , , and  as well as the complete set  are covers. So, four out of eight subsets are covers and, thus, at least half. Similarly, 4-vertex paths  (also known as P_4) have this property and so does any star like  or  (known as $K_{1,k}$ in general), while a 4-vertex cycle  (also known as C_4) does not, since out of the 16 possible subsets only seven are covers.

At first glance and from afar, this problem appears to be a rather hard problem since counting vertex covers is a hard problem in general: It is complete [33] for the class $\#P$, which is high up in the polynomial hierarchy. Of course, our task is not actually to determine the number $\#\text{vc}(G)$ of vertex covers of an arbitrary graph $G = (V, E)$, but “just” to determine whether $\#\text{vc}(G) \geq 2^{|V|}/2$ or $\#\text{vc}(G) < 2^{|V|}/2$ holds. Nevertheless, it seems, at least at first glance, that in order to determine whether this very fine and precise distinction holds, we need to compute $\#\text{vc}(G)$.

When we approach the problem more closely, we suddenly see that it is very much tractable: Indeed, in the earlier spirit of looking at a problem from different viewpoints, *seven different ways will be explored of proving that the problem can be solved efficiently*.

Before we have a closer look, some more rigorous definitions will be useful: As already indicated, an *undirected graph* G is a pair (V, E) of a *vertex set* V and an *edge set* E consisting of subsets of V of size 1 or 2 (so we allow loops, but no “empty edges”). A (*vertex*) *cover* of G is a set $C \subseteq V$ with $C \cap e \neq \emptyset$ for all $e \in E$. Let $\#\text{vc}(G)$ be the size of the set $\{C \subseteq V \mid C \text{ is a cover of } G\}$, so $\#\text{vc}(\triangle) = 4$ and $\#\text{vc}(\square) = 7$, and let $\text{Pr}_{\text{vc}}[G] := \#\text{vc}(G)/2^{|V|}$ be the fraction of subsets that are covers (or, equivalently, the probability that a randomly chosen subset is a cover), so $\text{Pr}_{\text{vc}}[\triangle] = \frac{1}{2}$ and $\text{Pr}_{\text{vc}}[\square] = \frac{7}{16}$. Then the graph problem whose complexity we study in this paper is:

Definition 1.1 (Majority of Vertex Covers Problem). $\text{MAJ-VC} := \{G \mid \Pr_{\text{vc}}[G] \geq \frac{1}{2}\}$.

The fact that covers $C \subseteq V$ intersect all edges of a graph clearly means that the remaining vertices $I = V \setminus C$ must form an independent set. Thus, MAJ-VC is the *same* problem (not just a different encoding, but literally the same problem) as asking whether at least half of all possible vertex sets are independent sets in a given graph.

A different way of looking at MAJ-VC is to think of the vertices of G as propositional variables and to think of a set $C \subseteq V$ as an assignment β that makes the variables in C true and all other variables false (formally, let $\mathbb{B} = \{\text{false}, \text{true}\}$ contain the two possible truth values and define $\beta: V \rightarrow \mathbb{B}$ by $\beta(v) = \text{true}$ for $v \in C$ and $\beta(v) = \text{false}$ for $v \notin C$). Then the “cover property,” by which $C \cap e \neq \emptyset$ must hold for all $e \in E$, means that for each $\{u, v\} \in E$ we must have $\beta(u) = \text{true}$ or $\beta(v) = \text{true}$. In other words, if we form a propositional formula ϕ_G that is a conjunction, taken over all $\{u, v\} \in E$, of the clauses $(u \vee v)$, then β must be a satisfying assignment of ϕ_G . For example, the graph  is the formal tuple $G = (\{v_1, v_2, v_3, v_4\}, \{\{v_1, v_2\}, \{v_2, v_3\}, \{v_1, v_3\}, \{v_3, v_4\}\})$ and it corresponds to the formula $\phi_G = (v_1 \vee v_2) \wedge (v_2 \vee v_3) \wedge (v_1 \vee v_3) \wedge (v_3 \vee v_4)$. Note that ϕ_G is in conjunctive normal form with at most two literals per clause and without negations; and also note that for any formula ϕ of this form there is a graph G such that $\phi = \phi_G$. Phrased differently, MAJ-VC can be seen as the problem of telling for a 2CNF formula ϕ without negations (a *monotone* formula) whether at least half of all assignments are satisfying. In the notation of [30], MAJ-VC would be written as $\text{MON-2SAT-PR}_{\geq 1/2}$.

Instead of switching to logic in order to generalize MAJ-VC , we can also stay in the context of graph theory, but consider *hypergraphs*. A k -*hypergraph* is a pair $H = (V, E)$ where V is still a vertex set and each $e \in E$ is a subset of V of size *at most* k . Thus, undirected graphs are 2-hypergraphs in this sense (ignoring the possibility of an empty edge $e = \emptyset$, which are allowed in hypergraphs, but forbidden in normal graphs). The generalization of vertex covers are *hitting sets*, which are sets $X \subseteq V$ such that $e \cap X \neq \emptyset$ holds for all $e \in E$. The problem of telling whether at least half of all vertex subsets of a, say, 3-hypergraph are hitting sets is then the problem MAJ-3HS . Note that this is the same problem as $\text{MON-3SAT-PR}_{\geq 1/2}$, so, once more, we can recast these graph problems as SAT problems.

Contributions of This Paper. As already mentioned, MAJ-VC , the problem of telling whether most vertex sets are covers of a graph, is a tractable problem – and readers are invited to try to come up with a polynomial-time algorithm at this point, if they have not already done so (it is not too hard, but certainly not trivial). The central contribution of the present paper is *not* showing that $\text{MAJ-VC} \in \text{P}$ holds: This follows easily from the work of Akmal and Williams [2], who show not only $\text{MAJ-VC} = \text{MON-2SAT-PR}_{\geq 1/2} \in \text{P}$, but even the *much* more general statement

$k\text{SAT-PR}_{\geq p} \in \text{P}$ for all $k \geq 1$ and all rational $p \in [0, 1]$, by which tractability still holds if we allow negated literals, if we allow clauses of any fixed size $k > 2$, if we allow other thresholds $p \neq 1/2$, and if we allow any combination thereof.

Rather, the present paper is about presenting seven different ways of proving the tractability of MAJ-VC (or, equivalently, of MON-2SAT-PR $_{\geq 1/2}$). We focus on this problem (and not on more general problems like the aforementioned $k\text{SAT-PR}_{\geq p}$) to keep the presentation simple (the analysis of $k\text{SAT-PR}_{\geq p}$ in [2] and also in [30] quickly becomes extremely technical), but also because some of the ideas presented in the following only work for monotone clauses or for size-2 clauses or both. Nevertheless, in each case we will have a look at how the presented ideas generalize (or do not).

The first way is quite straight(forward): Apply the *search tree technique* [10] from FPT theory. However, we need a small “twist” as our problem is not really a parameterized problem: Build a search tree by finding an arbitrary edge $\{u, v\} \in E$ in the graph and then add three numbers, namely the number of G ’s covers C with $C \cap \{u, v\} = \{u\}$, with $C \cap \{u, v\} = \{v\}$, and with $C \cap \{u, v\} = \{u, v\}$, which are obtained recursively. In a normal search tree, we end the recursion at a parameter-dependent depth since, when this depth is reached, “no solution” can exist; in our case, we end it with the answer “no majority of covers” when depth 3 is reached. We will show that this answer is, then, correct.

The second way is based on a technique from SAT solving: Backdoor sets [34]. We will see that whenever most vertex sets are covers, we can always find a backdoor set of size 4 into 1CNFS and this will allow us to compute the number of covers quite easily.

The third way of solving the problem is through randomization and is certainly the easiest to state in full: *On input G , randomly sample 2,250 subsets $C_1, \dots, C_{2,250} \subseteq V$ and claim “ $G \in \text{MAJ-VC}$ ” if at least 1,089 of them are covers.* While easy to state, deep insights into the *spectra of satisfaction probabilities of 2CNF formulas* will be needed to show that this algorithm gives a correct answer with probability at least 2/3, placing the problem in BPP, the randomized version of polynomial time (which will also count as “tractable”).

The fourth way leaves the solid path of concrete algorithmics and turns to algorithmic *meta* theorems. We will show that we can apply Courcelle’s Theorem [8] as follows: If the tree-width of G is at most 4, apply (a counting version of) Courcelle’s Theorem to the free second-order variable C in the formula $\forall x \forall y (x \sim y \rightarrow (C(x) \vee C(y)))$; and if the tree-width is larger, output “no majority of vertex covers.” While this application is superficially simple (at least for meta-theorem enthusiasts), intriguingly, it sadly does *not* generalize to, say, MAJ-3HS, meaning that we do not get a (much) simpler tractability result for that problem compared to the complex analysis in [2, 30].

The fifth way, which arguably leads us into an even more “abstract” landscape,

is to apply the famous graph minor theorem of Robertson and Seymour. By this theorem, if a property is “closed under taking minors” (a classical example is the class of planar graphs), then it can be characterized by a finite set of graphs that are “forbidden as minors” (in the case of planar graphs, by Kuratowski’s Theorem [19] the set is $\{\text{K}_5, \text{K}_{3,3}\}$, consisting of the 5-clique K_5 and the complete bipartite graph $K_{3,3}$). It is simple (but not trivial) to show that the property “most vertex sets are covers” is closed under taking minors and, hence, it can be characterized by a finite set of forbidden minors. Even better, we can explicitly state this set: It is $\{\text{II}, \text{I}; \text{I}\text{-}\text{I}, \text{II}\text{-}\text{II}, \text{I}\text{-}\text{II}, \text{II}\text{-}\text{I}\}$.

The sixth way replaces the powerful minors by forbidden induced subgraphs. This has the advantage of allowing us to generalize the approach (while the property “most vertex sets are covers” is closed under taking minors, this is no longer true for interesting variations), but has the disadvantage that the analogue of the Robertson–Seymour Theorem (“there is a finite set of forbidden induced subgraphs”) is not generally true. Nevertheless, we will see that for MAJ-VC we can restrict attention to tree-depth 5 and use this to solve MAJ-VC via a finite set of forbidden induced subgraphs.

The seventh way starts where the fifth and sixth way meet: We define a *well-quasi-ordering* on graphs (the graph minor relation is also a well-quasi-ordering) that is easy to decide and relative to which the property “most vertex sets are covers” is closed. The theory of well-quasi-orderings will then provide us with a set of forbidden elements that characterizes the property. In contrast to the subgraph relation and to the graph minor relation, the presented ordering is, on the one hand, a well-quasi-ordering and, on the other hand, lends itself to generalizations for problems like $k\text{SAT-PR}_{\geq p}$.

Related Work. As already indicated, MAJ-VC is a special case of counting–threshold problems for propositional formulas in conjunctive normal form (CNF formulas). These more general problems have, of course, attracted *a lot* of attention since already the basic problem SAT (“Is the number of satisfying assignments of ϕ at least 1?”) is NP-complete by the Cook–Levin Theorem [7, 20]. It remains NP-complete when the clauses are required to have size at most 3 (resulting in the 3SAT problem) and becomes NL-complete when the clause size is at most 2 (the 2SAT problem). The problem becomes trivial, of course, when no negations are allowed as, then, setting all variables to *true* is always a satisfying assignment.

When it comes to thresholds larger than 1 for the number of vertex covers of a graph or the number of satisfying assignments of a formula, the complexity landscape gets even more interesting. The mapping $\# : \text{CNFS} \rightarrow \mathbb{N}$ that maps CNF formulas ϕ to the number of their satisfying assignments is a classical #P-complete problem [33] and, hence, believed to be very hard. Furthermore, it remains

complete when we only allow only 3CNFs as input; and it *still* remains $\#P$ -complete when we only allow 2CNFs and *even still* when only monotone 2CNFs are allowed, see [33]. In other words, the function $\#\text{vc}(\cdot)$ considered in the present paper, which maps graphs to the number of their vertex covers, is a $\#P$ -complete function and so is $\Pr_{\text{vc}}[\cdot]$ since it is just a rescaled version.

When it comes to the “at most” or “majority” version, that is, to MAJ-VC or more generally $k\text{SAT-PR}_{\geq 1/2}$ or even $\text{SAT-PR}_{\geq 1/2}$, an astounding thing happens: While $\text{SAT-PR}_{\geq 1/2}$ is complete for PP (basically “a decision version of $\#P$ ” that is high up in the polynomial hierarchy by Toda’s Theorem [31]), Akmal and Williams [2] showed that $k\text{SAT-PR}_{\geq 1/2} \in \text{P}$ holds for all $k \geq 1$ (see also [30] for a simpler proof) and, thus, $\text{MAJ-VC} \in \text{P}$. It is noteworthy that Akmal and Williams start their discussion of $k\text{SAT-PR}_{\geq 1/2}$ with the case $k = 2$ as an introductory example and one of the “ways” presented in the present paper (namely the use of backdoor sets) is also sketched in their paper.

The basics of the theoretical backgrounds for the seven ways of proving the tractability of MAJ-VC will be sketched in the main text, but for readers interested in more details, here are some possible entry points to the literature: Concerning fixed-parameter tractability theory, two classical textbooks are [10, 13]; for a gentle introduction to algorithmic meta-theorems, see for instance [29]; for more background on well-quasi-orderings, see [18]. Concerning the ubiquitous technique of randomization, finding an “entry point” to the literature seems hopeless, but in the particular case of finding satisfying assignments for $k\text{CNF}$ formulas, readers should have a look at Schöning’s ingenious use [28] of randomization to solve the $k\text{SAT}$ problem at some point (although [23] shows that randomization is not *actually* necessary).

Organization of This Paper. The paper will take you on a tour via seven ways, each of which is described in one of the following sections, through a fascinating landscape of complexity and algorithms. Each section starts with a bit of “scenery description,” followed by background on the method or tools employed (like search trees or well-quasi-orderings and so on), followed by a theorem stating and a proof showing the tractability of MAJ-VC . Each section then finishes with some observations concerning whether or not the way can be generalized.

2 The First Way: Via Search Trees

As you start your tour through the complexity landscape surrounding MAJ-VC , the first thing you notice is, of course, that there are lots of trees: Computer Science in general, and Theoretical Computer Science in particular, is a land of lush forests filled with trees of all kinds. The first way soon leads you to a fenced-off area with

majestic trees growing in it. Next to an inviting entrance gate, a gardener greets you: “Welcome to the Search Tree Nursery. We grow – and sell – all kinds of *Quaerere arbore!* Can I interest you in any of our trees? And, of course, it would be helpful to know a bit about your budget.” You respond that you need a tree for a counting–threshold problem and that your budget is polynomially bounded. The gardener seems a bit at a loss upon hearing about your budget (apparently, polynomial time is considered a *very* tight budget in the Search Tree Nursery), but then her eyes light up: “That is an uncommon request, I dare say, but I like a challenge. You see, most people use search trees for, well, searching, not for counting and certainly not for counting–threshold frivolities. And your budget *is* restricted. But do not fear, I think I have just the right tree for you.” She leads you to a part of the nursery where a sign says “Bonsai and assorted $O^*(1)$ ” and then hands you a small, but beautiful specimen of *Quaerere arbore*.

We will examine this particular specimen more closely in a moment, but let us first have a brief closer look at the vertex cover problem.

Background on the Vertex Cover Problem in FPT Theory. The vertex cover problem is the poster child of FPT theory, the theory of fixed-parameter tractability, as the core ideas of this theory (search trees and kernelization, in particular) work spectacularly well for it (see [15] for a recent record). However, the parameterized problem p-vc (or, in full, p-VERTEX-COVER) studied in this theory is fundamentally different from the counting–threshold problem MAJ-vc we study in the present paper: For p-vc the input is a pair (G, k) of an undirected graph G and a natural number $k \in \mathbb{N}$ (the *parameter* in FPT parlance) and the question is whether there is a cover C of G of size $|C| \leq k$. The problems are connected, nevertheless as, intuitively, if a graph has “many” covers, there must be some “small” covers among them. The following lemma shows that this intuition is very much correct:

Lemma 2.1. *Let $G = (V, E)$ be a graph with $\Pr_{vc}[G] \geq 1/2$. Then all matchings in G have size at most 2 and G has a vertex cover of size at most 4.*

Proof. Suppose G contains a matching of size 3 ($\bullet\bullet\bullet = M_3$) as a subgraph, that is, there are six different vertices $v_1, \dots, v_6 \in V$ with $\{\{v_1, v_2\}, \{v_3, v_4\}, \{v_5, v_6\}\} \subseteq E$. Then the probability that a random subset $C \subseteq V$ covers the edge $\{v_1, v_2\}$ is $3/4$ and the same is true for the other edges. Since these probabilities are independent, the probability that all three edges are covered is at most $(\frac{3}{4})^3 = \frac{27}{64} < 1/2$. Hence, the probability $\Pr_{vc}[G]$ that *all* edges of G are covered is less than $1/2$. For the second claim, consider a maximal matching $M \subseteq E$. Then $|M| \leq 2$ and the at most 4 vertices in M (that is, in $\bigcup M$) form a (vertex) cover of G since, otherwise, there would be an uncovered edge in E and the matching would not be maximal. \square

The reverse direction of the lemma's statement is not true: A graph consisting of two large stars (like ) can be covered by just two vertices (), but just a little more than one quarter of all vertex sets are covers. This means that we will have to put forward some extra ideas to make the *search tree technique*, which works so well for the p-vc problem, also work for MAJ-vc.

Background on the Search Tree Method. Suppose we are given a pair (G, k) and wish to determine whether there is a cover $C \subseteq V$ of G of size $|C| \leq k$. If $k = 0$, the answer is obviously “no” unless $E = \emptyset$. The other way round, if $E = \emptyset$, the answer is always “yes.” The interesting case is $k > 0$ and $E \neq \emptyset$. Then there must be an edge $\{u, v\} \in E$. The key insight behind the search tree method is that *for every cover C of G we must have $u \in C$ or $v \in C$* . Admittedly, when written down like this, the insight feels utterly trivial, but fear not: Theoretician have found ways to complicate the idea beyond all recognition. Anyway, in the case $u \in C$, the vertex u will cover all edges $e \in E$ that contain u , but $C \setminus \{u\}$ must then cover “the rest” of the edges, that is, the edges in $E \setminus \{e \in E \mid u \in e\}$. (The situation for the second case is symmetric, with u and v exchanged.) In particular, we can only cover G with a size- k cover, if we can cover either $G \setminus \{u\}$ or $G \setminus \{v\}$ with a size- $(k - 1)$ cover (here, $G \setminus X := (V \setminus X, E \cap \{\{u, v\} \mid u, v \in V \setminus X\})$ results from G by removing all vertices in X together with any adjacent edges) – and we can then, of course, try to answer these questions recursively, see Figure 1 for an example. The resulting recursion tree is called the *search tree*, hence the name of the method. Note that its depth is at most k (we stopped the recursion at $k = 0$) and we have two recursive calls in the algorithm, resulting in two children at each inner node of the search tree. All told, the search tree has size 2^k and the runtime of the algorithm will be something like $O(2^k|E|)$, depending a bit on which data structures we use for representing E .

An advanced version of this idea, which will be useful in our counting-threshold setting in a moment, is that instead of branching to the two cases “ C contains u ” and “ C contains v ”, we cleverly choose the edge $\{u, v\} \in E$ and then branch to *three* cases, namely “ C contains u , but not v ” and “ C contains v , but not u ” and “ C contains both u and v .” The insight is that in the branch that looks for covers that “contain u , but not v ,” we know that when v is not part of a cover, *all neighbors of v must be part of the cover*, allowing us to remove all edges adjacent to these neighbors and to look for a much smaller cover for the rest, see Figure 2 for an example. By choosing the edge $\{u, v\} \in E$ to maximize the number of these neighbors, we can ensure that while there will always be three branches, the subtrees will have only half the depth, leading to a better overall runtime. Of course, we can try to take this idea further and cleverly choose more than a single edge and branch more broadly to get even more shallow trees. A small cottage

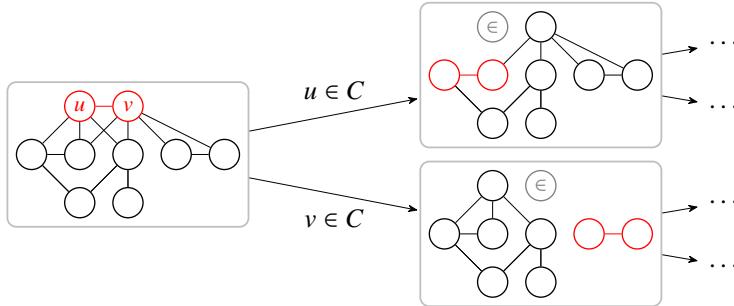


Figure 1: A binary search tree for finding a vertex cover of size k in an undirected graph $G = (V, E)$: At each node of the tree, starting with the graph itself at the root, shown left, an edge $\{u, v\} \in E$ is selected (shown in red). Then any vertex cover C of G must contain either u , resulting in the graph $G \setminus \{u\}$ shown above (with u no longer being a vertex of the graph), or v , resulting in $G \setminus \{v\}$. Note how, in either case, we have identified one vertex of the cover, meaning that the rest of the graph must be covered with $k - 1$ vertices.

industry has sprung up around this method, resulting in improved search tree sizes (and, hence, improved runtimes [15]) for the vertex cover problem and related problems, but for our purposes the “branching to $\{u, v\} \cap C = \{u\}$, $\{u, v\} \cap C = \{v\}$, and $\{u, v\} \cap C = \{u, v\}$ ” will suffice.

Applying the Search Tree Method. Adapting the search tree method so that we *count* the number of vertex covers of a graph, is not very hard: Instead of just answering “yes” or “no,” we let the algorithm output the number of vertex covers it has found. For instance, when $E = \emptyset$ holds, instead of just answering “yes,” we answer $2^{|V|}$, which is the number of vertex covers that an empty graph on $|V|$ vertices has. Crucially, when we recursively branch for the three cases $\{u, v\} \cap C = \{u\}$, $\{u, v\} \cap C = \{v\}$, and $\{u, v\} \cap C = \{u, v\}$, the total number of covers will be the *sum* of the three numbers computed – and note that in this three-way branching we do not double-count any covers (which we would, if we just branched to the cases $u \in C$ and $v \in C$).

What about the depth of the search tree? Usually, we stop the search and get a leaf node in the tree when we know that “any cover below this node” (meaning, any cover satisfying all the requirements imposed by the nodes on the path from the root) will have size greater than k . In our case, however, we clearly cannot ignore these covers as most covers will be large. The trick is to use Lemma 2.1: Each time we recurse, the chosen edge will be completely disjoint from the previously chosen edges. By the lemma, if we still find an edge at recursion depth 3, we can

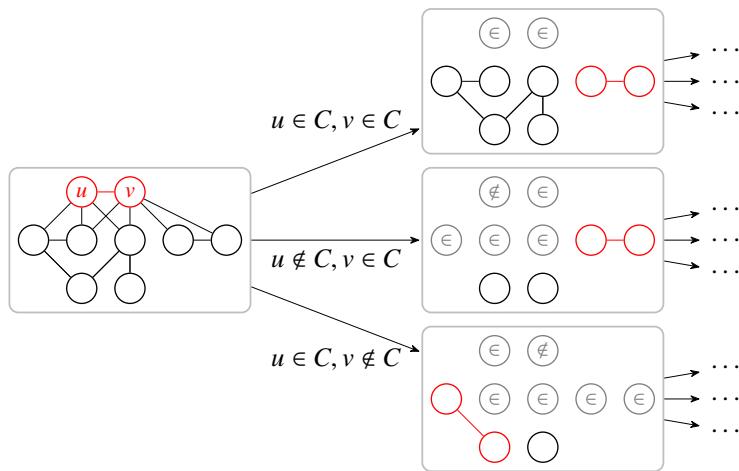


Figure 2: A ternary search tree for counting the number of vertex covers of an undirected graph $G = (V, E)$. In contrast to the recursion in Figure 1, at each node we classify the possible covers C of G according to whether $C \cap \{u, v\} = \{u, v\}$ holds (top case) or $C \cap \{u, v\} = \{u\}$ or $C \cap \{u, v\} = \{v\}$ (the other two cases). In the lower cases, $u \notin C$ (or $v \notin C$) forces the neighbours of u (or v) to be part of the cover. Crucially, the number of covers that the graphs in the branches have (39 for the top graph, 12 for the middle graph, 6 for the lower graph) add up to the number of covers the root graph has (57 out of 512 possible vertex sets).

break off the whole computation.

Algorithm 1: An algorithm based on search trees for deciding whether an undirected input graph $G = (V, E)$ is an element of MAJ-vc. In the algorithm, $N[u] := N(u) \cup \{u\}$ is the closed neighbourhood of u and $N(u) = \{w \in V \mid \{u, w\} \in E\}$ is the (open) neighbourhood. For the correctness of the algorithm, see the proof of Theorem 2.2.

```

1 input  $G$ 
2 if  $\text{count-vc}(G, 0) \geq 2^{|V|}/2$  then output “ $G \in \text{MAJ-vc}$ ” else output “ $G \notin \text{MAJ-vc}$ ”
3
4 function  $\text{count-vc}(G = (V, E), d)$  //  $d$  is the size of an already found matching
5   if  $E = \emptyset$  then return  $2^{|V|}$  // every subset is a cover
6   if  $d \geq 2$  then output “ $G \notin \text{MAJ-vc}$ ” and halt // see the proof for the correctness
7   if  $\{v\} \in E$  for some  $v \in V$  then // self-loop at  $v$ 
8     return  $\text{count-vc}(G \setminus \{v\}, d + 1)$ 
9   else
10    { $u, v\} \leftarrow \text{an arbitrary element of } E$ 
11     $c_1 \leftarrow \text{count-vc}(G \setminus N[u], d + 1)$ 
12     $c_2 \leftarrow \text{count-vc}(G \setminus N[v], d + 1)$ 
13     $c_3 \leftarrow \text{count-vc}(G \setminus \{u, v\}, d + 1)$ 
14    return  $c_1 + c_2 + c_3$ 
```

Theorem 2.2. *Algorithm 1 decides MAJ-vc in time $O(|E|)$.*

Proof. The algorithm is a simple recursion, in which the function $\text{count-vc}(G, d)$ returns the number of vertex covers of G and the number d is the size of an already found matching (and also the depth of the recursion). Clearly, if we can show that the implementation correctly computes this function, the overall algorithm is correct.

If $E = \emptyset$ holds in line 5, every subset $C \subseteq V$ is a cover of G , so $2^{|V|}$ is the correct return value. If this is not the case, there must be an edge in E . However, if $d \geq 2$ holds, we know that this edge would constitute the third edge in a matching in E and by Lemma 2.1 we correctly output “ $G \notin \text{MAJ-vc}$ ” in this case (and halt the whole computation, not just this branch of the recursion). Otherwise, we check whether there is a vertex v with a loop, in which case every cover *must* contain v , so the total number of covers of G equals the number of covers of $G \setminus \{v\}$, which results from G by removing v and all edges containing v . In other words, the value returned in line 8 is correct.

Finally, we consider a size-2 edge $\{u, v\} \in E$. The three values c_1 , c_2 , and c_3 are now exactly the number of covers C of G such that $C \cap \{u, v\} = \{v\}$, $C \cap \{u, v\} = \{u\}$, and $C \cap \{u, v\} = \{u, v\}$, respectively. To see this, just note that $C \cap \{u, v\} = \{v\}$ means that u is *not* in the cover and, hence, all neighbours of u must be in the cover.

In other words, for every cover C' of $G \setminus N[u]$, the set $C' \cup N(u) = C \cup \{w \in V \mid \{u, w\} \in E\}$ is a cover C of G with $C \cap \{u, v\} = \{v\}$. By a similar argument, we get that the numbers c_2 and c_3 are correct, meaning that the final output $c_1 + c_2 + c_3$ is also correct.

For the runtime, just observe that because of the cap on the recursion depth, the search tree has size at most $3^2 = 9$. All other computations take time linear in $|E|$ assuming appropriate data structures. \square

Extensions. Given that the search tree method is a rather powerful and generic tool, it should come as no big surprise that we can use it to solve not only **MAJ-VC**, but also more general problems:

- The threshold “ $1/2$ ” used in the definition of **MAJ-VC** for the fraction of vertex sets that must be covers is both natural and a bit arbitrary (why not “ $1/3$ ” or “ $3/4$ ”?). If we change the threshold from $1/2$ to a different number $p \in [0, 1]$, we get the problem **MON-2SAT-PR $_{\geq p}$** (see the introduction). Having a look at the proof of Lemma 2.1, we see immediately that the following generalization holds:

Lemma 2.3. *Let G be a graph with $\Pr_{vc}[G] \geq p > 0$. Then all matchings in G have size at most $\log_{3/4} p$ and G has a vertex cover of size at most $2 \log_{3/4} p$.*

This means that Theorem 2.2 still works if we replace “ $\text{count-vc}(G, 0) \geq 2^{|V|}/2$ ” by “ $\text{count-vc}(G, 0) \geq 2^{|V|} \cdot p$ ” in line 2 and “ $d \geq 2$ ” by “ $d \geq \log_{3/4} p$ ” in line 6. Observe that while the size of the search tree is constant for any *fixed* p , for arbitrary p , the size of the search tree and hence also the runtime is $3^{\log_{4/3}(1/p)}$ and hence exponential in the number of bits of $\lceil 1/p \rceil$ – and this is no coincidence: A polynomial dependence of the runtime on the number of bits of $\lceil 1/p \rceil$ would allow us to compute $\Pr_{vc}[G]$ and hence also $\#\text{vc}(G)$ in polynomial time using binary search, but $\#\text{vc}(\cdot)$ is a #P-complete function.

- We can modify the test from line 2 further and check whether we have “ $\text{count-vc}(G, 0) > 2^{|V|} \cdot p$ ” (rather than “ $\dots \geq \dots$ ”) and get an algorithm for solving **MON-2SAT-PR $_{>p}$** in polynomial time. The difference may seem slight, but as first observed by Akmal and Williams in [2] and then studied in more detail in [30], for counting-threshold problems it can make a *huge* difference whether “ $\geq p$ ” or “ $> p$ ” is considered.
- It is not too hard – but no longer a trivial code modification – to extend Algorithm 1 to work for the more general **2SAT-PR $_{\geq p}$** problem, where negated literals are allowed. It will, however, be more natural to look at this problem in the context of backdoor sets, which we do in the next section.

- Concerning the even more general problem $\text{MAJ-3HS} = \text{MON-3SAT-PR}_{\geq 1/2}$, let alone $k\text{SAT-PR}_{\geq p}$ for arbitrary $k > 2$ and p , the search tree method (at least, as presented) simply fails: It is not hard to construct 3-hypergraphs for which there is no way to “stop early in line 6”. For instance, the hypergraph H with the edges $\{a, x_1, y_1\}, \{a, x_2, y_2\}, \dots, \{a, x_m, y_m\}$ is covered by every vertex set containing a (so $\Pr_{3\text{HS}}[H] > 1/2$), but the search tree would have depth m .

3 The Second Way: Via Backdoor Sets

Leaving the Search Tree Nursery behind you, you continue on your journey through the landscape on a second way that leads you further through hills and forests. Just as you exit a small valley, an impressive vista opens before you and reveals a view of a huge mountain in the distance. It takes you quite a while to get to its foot, where you meet a courteous hobbit who seems to be waiting for someone. He immediately addresses you: “Well met, Master Traveller. What brings you to the Lonely Mountain? The treasures inside, I would venture. Well, I, for one, am waiting for a group of dwarves and a wizard, but they seem to be *quite* late, I must say.” You are about to inquire about the treasures, but the hobbit just keeps talking: “I, for one, *really* do not agree with the dragon that the gold is the real treasure. He really should get out more, I *keep* telling him. Anyway, in this gentlehobbit’s opinion, the real treasure is the machinery! All these wonderful devices for *counting* the gold and what-have-you.” Your interest sparks at this, but before you can squeeze in a question about counting machinery, the oration continues relentlessly: “Of course, getting in is out of the question, this mountain is a real fortress: As you can see, the old dwarven entrance here is *very* securely locked down. And, of course, all the little windows further up and the tunnels on the sides are barred by large iron grilles. I guess, one might be able to use one of the chimneys that come out at the mountain top, but you would have to be an *exceptionally* skilled climber just to get up there, let alone then getting down them. So, getting in is totally out of the question. Unless, of course...” Now the hobbit stops talking, obviously expecting you to ask him to elaborate. Since you would very much like to get into the mountain, preferably avoiding the dragon, you coax him to continue, which he happily does: “You see, I happen to have this map with a secret *backdoor* marked on it. A bit difficult to read, if I may say so, but I can lend you my moonshine lamp. I was going to give the map to the dwarves, but as I told you, they are *quite* late. So, here, you take it and go inside and have a look around!”

As we will see in the following, the backdoor does, indeed, grant us access to MAJ-VC .

Recall that MAJ-vc is a special case of the problem $2\text{SAT-PR}_{\geq 1/2}$, where the input

is a propositional formula ϕ in conjunctive normal form with at most two literals per clause (a 2CNF formula) and the question is whether at least half of all possible assignments $\beta: \text{vars}(\phi) \rightarrow \mathbb{B}$ satisfy ϕ (with $\text{vars}(\phi)$ of course denoting the set of variables in ϕ and $\mathbb{B} = \{\text{false}, \text{true}\}$ denoting the two possible truth values): Covering all edges $\{u, v\} \in E$ corresponds exactly to satisfying all clauses of the form $(u \vee v)$. For example, the number of vertex covers of $G = \bullet\bullet\bullet$ is exactly the number of satisfying assignments of $\phi_G = (v_1 \vee v_2) \wedge (v_1 \vee v_3) \wedge (v_2 \vee v_3) \wedge (v_3 \vee v_4)$. It is, thus, no surprise that the tools that have been developed for solving SAT problems also apply to MAJ-vc. One such tool are backdoor sets [34].

Background on Backdoor Sets. The paramount goal of SAT solving is to determine on input of a formula ϕ in conjunctive normal form whether it has a satisfying assignment. Since SAT is the NP-complete problem *par excellence*, we cannot really hope to devise an efficient algorithm for solving this problem for arbitrary ϕ , but if ϕ has a special form and is sufficiently “simple,” even highly efficient algorithms are known. For instance, if $\phi \in \text{2CNFS}$ holds (that is, when ϕ is in conjunctive normal form with at most two literals per clause), we can decide whether ϕ is satisfiable in polynomial time (more precisely, the 2SAT problem is NL-complete), and this is also true when $\phi \in \text{HORN-CNF}$ holds (meaning that there is at most one positive literal per clause) as HORN-SAT is P-complete [22].

The idea behind backdoor sets is that while a CNF formula ϕ may not be syntactically simple in one of the above senses, it may be “near” to such a formula as only a small number of variables cause a deviation. For instance, $\phi \notin \text{2CNFS}$ might hold, but there might be only, say, four variables x_1, x_2, x_3 , and x_4 (which will be called a *backdoor set* in a moment) such that removing them from the formula would result in an element of 2CNFS. The key observation is that there actually is a simple way of “removing” variables from any CNF formula ϕ : Iterate over all possible assignments to the variables in the backdoor set and then use unit propagation to remove the variables. These ideas are detailed in the following two definitions:

Definition 3.1. Let ϕ be a CNF formula, let X be some variables, and let $\beta: X \rightarrow \mathbb{B}$ be a function (a “partial assignment”). Define $\phi|_\beta$ as the formula obtained from ϕ by

1. removing all clauses from ϕ containing a literal made true by β , that is, containing a variable $v \in X$ with $\beta(v) = \text{true}$ or containing $\neg v$ for some $v \in X$ with $\beta(v) = \text{false}$; and
2. removing from the remaining clauses all literals containing a variable from X .

As an example, for $\phi = (x \vee \neg y \vee a) \wedge (\neg x \vee b \vee \neg c) \wedge (y \vee d) \wedge (d)$ and $X = \{x, y\}$ and $\beta(x) = \text{true}$ and $\beta(y) = \text{false}$, we have $\phi|_\beta = (\cancel{x} \vee \cancel{y} \vee a) \wedge (\neg \cancel{x} \vee b \vee \neg c) \wedge$

$(\bar{x} \vee d) \wedge (d) = (b \vee \neg c) \wedge (d)$. Note how $\phi \notin 2\text{CNFS}$ holds, while $\phi|_\beta \in 2\text{CNFS}$ holds both for our particular β and also for the three other possible $\beta: X \rightarrow \mathbb{B}$.

Definition 3.2. Let Ψ be a set of CNF formulas and let ϕ be any CNF formula. A (*strong*) backdoor set into Ψ for ϕ is a set X of variables such that for all $\beta: X \rightarrow \mathbb{B}$ we have $\phi|_\beta \in \Psi$.

To get an intuition for the above definition, consider a formula ϕ , a fixed set $V \supseteq \text{vars}(\phi)$ of variables, and an arbitrary set $X \subseteq V$. If we restrict an assignment $\alpha: V \rightarrow \mathbb{B}$ that satisfies ϕ to just the variables in $V \setminus X$, we get a satisfying assignment α' of $\phi|_\beta$, where β is the restriction of α to X . The other way round, we can extend any satisfying assignment α' of $\phi|_\beta$ to one of ϕ by joining α' with β . In particular, if we write $\#(\phi)$ for the number of satisfying assignments that ϕ has relative to V , we get (the division by $2^{|X|}$ is needed as $\#(\phi|_\beta)$ is also counted relative to the original V):

$$\#(\phi) = \sum_{\beta: X \rightarrow \mathbb{B}} \frac{\#(\phi|_\beta)}{2^{|X|}}. \quad (1)$$

For instance, for $V = \{a, b, c, d\}$ and $\phi = (a \vee \neg b) \wedge (\neg a \vee b \vee c)$ and $X = \{a\}$, we have: $\#(\phi) = 10$ as ϕ has ten satisfying assignments $\alpha: \{a, b, c, d\} \rightarrow \mathbb{B}$; for $\beta_0(a) = \text{false}$ we have $\phi|_{\beta_0} = (\neg b)$ and $\#(\phi|_{\beta_0}) = 8$ as $(\neg b)$ has eight satisfying assignments $\alpha: \{a, b, c, d\} \rightarrow \mathbb{B}$; for $\beta_1(a) = \text{true}$ we have $\phi|_{\beta_1} = (b \vee c)$ and $\#(\phi|_{\beta_1}) = 12$; and, indeed, $10 = (8 + 12)/2^{|X|} = 20/2$.

The crucial property of equation (1) is that $\phi \in \text{SAT}$ holds (the left hand side is positive) iff $\phi|_\beta \in \text{SAT}$ holds for at least one $\beta: X \rightarrow \mathbb{B}$ (at least one summand in the sum on the right hand side is positive). Now, if X is a small backdoor set into, say, $\Psi = 2\text{CNFS}$, deciding $\phi \in \text{SAT}$ becomes easy: We just have to run $2^{|X|}$ many test of the form “Is this 2CNF formula satisfiable?”

Of course, many formulas will not have a small backdoor set and, even when they do, it may be difficult to find. A rich theory has been developed (see for instance [11, 21] for some recent results) that tries to deal with these and other problems, but the simple above form of backdoor sets will suffice for solving MAJ-VC.

Applying Backdoor Sets. Recall from Lemma 2.1 that all $G \in \text{MAJ-VC}$ have a vertex cover of size 4 or less. Let $X = \{v_1, v_2, v_3, v_4\}$ be such a small cover and now consider what $\phi_G|_\beta$ must look like: Since X is a cover, in each clause $(u \vee v)$ of ϕ_G at least one of the two variables is present in X and the clause is either completely removed in $\phi_G|_\beta$ or one of the variables is removed. Thus, all clauses of $\phi_G|_\beta$ have size at most 1 (see Figure 3 for an example) and it is trivial to compute the number of satisfying assignments that $\phi_G|_\beta$ has. Equation (1) then tells us how to compute the total number of satisfying assignments of ϕ . The details follow.

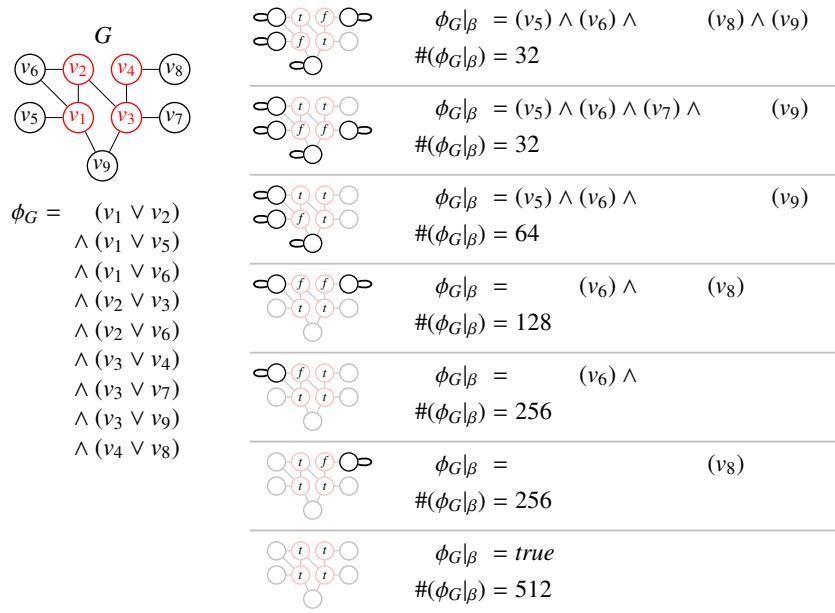


Figure 3: A graph in which $X = \{v_1, \dots, v_4\}$ forms a vertex cover and, thus, these variables form a backdoor set into 1CNFS for ϕ_G . Out of the sixteen possible $\beta: X \rightarrow \mathbb{B}$ only those are indicated that do not produce an empty clause in $\phi_G|_\beta$ (meaning that some internal edge of the X -set is not covered). For them, the resulting formula $\phi_G|_\beta$ is given and $\#(\phi_G|_\beta)$ relative to the original size-9 vertex set $V = \{v_1, \dots, v_9\}$. By equation (1), we get $\#(\phi) = (32 + 32 + 64 + 128 + 256 + 256 + 512)/16 = 80$.

Algorithm 2: An algorithm based on backdoor sets for deciding whether an undirected input graph $G = (V, E)$ is an element of MAJ-vc. Since Lemma 2.1 tells us such G cannot contain a matching of size 3, there will be at most four vertices in the backdoor set X in 1CNFS.

```

1 input  $G$ 
2
3  $M \leftarrow$  greedily compute a matching in  $G$  that is maximal or has size 3
4 if  $|M| > 2$  then output “ $G \notin \text{MAJ-vc}$ ” and halt // correct by Lemma 2.1
5  $X \leftarrow \bigcup M$  // the at most 4 vertices in the matching
6
7  $\phi_G \leftarrow$  the 2CNF formula representing  $G$ 
8  $c \leftarrow 0$  // counter for the number of covers
9 foreach  $\beta: X \rightarrow \mathbb{B}$  do // at most 16 possible  $\beta$ 
10    $\psi \leftarrow \phi_G|_\beta$  // a shorthand
11   if  $\psi$  contains no empty clause and not both a clause  $(v)$  as well as  $(\neg v)$  then
12      $c \leftarrow c + 2^{|V|-|\text{vars}(\psi)|}/2^{|X|}$ 
13
14 if  $c \geq 2^{|V|}/2$  then output “ $G \in \text{MAJ-vc}$ ” else output “ $G \notin \text{MAJ-vc}$ ”
```

Theorem 3.3. Algorithm 2 decides MAJ-vc in time $O(|E|)$.

Proof. The algorithm first greedily computes a maximal matching M in G , but we can actually stop this computation early when $|M|$ exceeds 2 since, by Lemma 2.1, we then know that $G \notin \text{MAJ-vc}$ holds and stop in line 4. Otherwise, the set $X = \bigcup M$ of vertices in M can have size at most 4 and is a vertex cover of G . Crucially, this implies that X is a backdoor set for ϕ_G into 1CNFS: The clauses of ϕ_G are exactly of the form $(u \vee v)$ for $\{u, v\} \in E$ and the fact that X is a cover of G means that in $\phi_G|_\beta$ every such clause is either missing completely or missing one variable. This means that by equation (1), the main loop correctly computes $\#(\phi_G) = \#\text{vc}(G)$ provided that we correctly sum up the values of $\#(\phi_G|_\beta)$. However, for a 1CNF formula ψ , the number of satisfying assignments relative to a fixed size- n set of variables is either 0 (when it is a contradiction, which we rule out by the checks in line 11) or is 2 raised to the number of variables *not* mentioned in ψ .

All told, the output at the end of the algorithm is correct and can be computed easily in time linear in $|E|$ assuming appropriate data structures since the main loop iterates over at most 16 different β . \square

Extensions.

- While Algorithm 2 ostensibly solves MAJ-vc, it is really an algorithm for deciding 2SAT-PR $_{\geq 1/2}$ in thinly veiled disguise: Just take a formula $\phi \in \text{2CNFS}$

as input instead of constructing it from a graph G and replace the notion of a “matching” by “a set of variable-disjoint clauses.” The rest of the algorithm is already stated in a way so that it also works in the presence of negations (namely in line 11, where we test the presence of contradictory clause pairs (v) and $(\neg v)$, which actually cannot arise when ϕ_G contains no negations).

- Just as in Algorithm 1, we can easily adapt the algorithm to work for $2\text{SAT-PR}_{\geq p}$ for arbitrary $p \in [0, 1]$ by replacing the test “ $M > 2$ ” by “ $M > \log_{3/4} p$ ” in line 4 and adapting the final line of the algorithm. An adaption of the last line is also all that is needed to decide $2\text{SAT-PR}_{>p}$.
- When we try to extend the algorithm to 3CNF formulas ϕ in order to solve $3\text{SAT-PR}_{\geq 1/2}$, we get further than we did on the first way via search trees, but still run into a roadblock: It is not too hard to see that when at least half of all assignments of $\phi \in 3\text{CNFS}$ are satisfying, there must still be a small backdoor set, but now into 2CNF formulas rather than 1CNF formulas. Unfortunately, recall that it is #P-complete to compute the number of satisfying assignments of 2CNF formulas, so we cannot simply use a recursion in the main loop.

4 The Third Way: Via Random Sampling

You backtrack out of the mountain, return the map and the moonshine lamp to the hobbit, and thank him once more for showing you the backdoor. Then you are back to hiking, following a third way. As the Lonely Mountain gets lost in the distance behind you, on the side of the road you come by two large silos with the sign *Theoretical Foods Inc.* prominently displayed on both of them. An agitated company employee sees your approach and immediately seeks your help: “Can you *please* help me? You see, we produce two delicious rice mixtures consisting of white rice and wild rice (whose grains are black). The mixtures differ only slightly in the composition: In one there are at least as many black grains as white ones, while in the other there are less black than white ones. Two deliveries have just been made, one of each mixture, and we placed them in two silos, each holding one billion grains. Unfortunately, it is no longer clear which mixture was put in which silo!” Counting all the grains in either silo is clearly hopeless, but you come up with a simple test: From the mixture in the first silo, scoop up a handful of grains (say, a thousand) and count (just) them. Intuitively, if there are much fewer black grains in your scoop than white ones (say, 400 black to 600 white), it is (highly) unlikely that the total number of black grains in the silo is larger than that of the white ones. Unfortunately, in your scoop there are exactly 498 black and 502 white grains – and you really cannot tell whether this really means that are less black grains overall than white ones.

The connection to MAJ-vc is, of course, that while for a large graph G it is hopeless to examine all vertex subsets of G (there are simply too many of them), if we “randomly sample a scoop of them” we can easily count the covers (the “black” grains) and the non-covers (the “white” grains). If there are many more covers in the sample than non-covers, we can be fairly sure that $G \in \text{MAJ-vc}$ holds, and if there are many more non-covers than covers, then $G \notin \text{MAJ-vc}$ is very likely. The tricky part is, as for the rice grains, the case that the numbers are the same or almost the same in our sample: What does that mean for the “overall mixture of covers and non-covers”?

You are about to tell the employee that, sadly, your idea does not seem to work, when they suddenly volunteer additional information: “You know, in our mixture with less black grains than white grains, we actually put in only at most 46.875% black grains (rather than 49.999%). Does that help in any way?” You soon realize that it sure does: In your scoop of 1,000 grains with 498 black ones and 502 white ones, you are now pretty confident that there are actually at least as many black grains as white grains overall. Otherwise, you would expect to find only 469 black grains, but have found 498 – a strong deviation.

The connection to MAJ-vc is now rather surprising: It turns out that by the *Spectral Well-Ordering Theorem* [30] the “mixture of covers and non-covers of graphs” has the peculiar property that for any graph G , the fraction $\Pr_{\text{vc}}[G]$ of covers is either at least $\frac{1}{2}$ or at most $\frac{15}{32} = 46.875\%$, exactly as for the mixtures of Theoretical Foods Inc. In other words, the *spectrum of values that $\Pr_{\text{vc}}[G]$ can have* has a gap in the interval $(\frac{15}{32}, \frac{1}{2})$.

In the following, let us have a closer look at the math behind the sampling technique and these peculiar spectra.

Background on Random Sampling. The mathematical analysis of random sampling (here in the form of a simple Bernoulli process) dates back hundreds of years and is among the best-studied topics of statistics and perhaps mathematics in general. Accordingly, our simple problem of counting how many randomly chosen vertex subsets are covers, is very well-understood from a statistical point of view. In detail, given an undirected graph $G = (V, E)$, if we pick $C \subseteq V$ uniformly at random, then by definition $p := \Pr_{\text{vc}}[G]$ is the probability that C is a cover of G . Sampling t different $C_1, \dots, C_t \subseteq V$ independently and uniformly at random and checking for each whether it is a cover of G yields a Bernoulli process for the probability p . The expected number of C_i that are covers is clearly $p \cdot t$, and bounding the probability that we deviate strongly from this value is well-studied:

Fact 4.1 (Chernoff–Hoeffding Bound [16]). *Let Z_1, \dots, Z_t be independent random variables, each taking value 1 with probability p and 0 with probability $1 - p$. Let*

$\epsilon > 0$. Then

$$\Pr\left[\frac{1}{t} \sum_{i=1}^t Z_i \geq p + \epsilon\right] \leq \left(\left(\frac{p}{p+\epsilon}\right)^{p+\epsilon} \left(\frac{1-p}{1-p-\epsilon}\right)^{1-p-\epsilon}\right)^t, \quad (2)$$

$$\Pr\left[\frac{1}{t} \sum_{i=1}^t Z_i \leq p - \epsilon\right] \leq \left(\left(\frac{p}{p-\epsilon}\right)^{p-\epsilon} \left(\frac{1-p}{1-p+\epsilon}\right)^{1-p+\epsilon}\right)^t. \quad (3)$$

In our setting, $Z_i = 1$ when C_i is a cover of G and $Z_i = 0$ otherwise, so $\frac{1}{t} \sum_{i=1}^t Z_i$ is exactly the fraction of C_i that are covers. For, say, $p = 1/2$ and $\epsilon = 1/64$, the theorem tells us that the probability that this fraction deviates from the expected value $1/2$ by more than $1/64$ is at most b^t for the constant $b \approx 0.9995117584$. While this constant is quite close to 1, it is strictly less than 1 and b^t will start to tend to 0 quickly for larger t . Indeed, for $t = 2,250$ we have $b^t < 1/3$. Figure 5 on page 26 visualizes this and similar situations.

Random sampling has also become an indispensable tool in complexity theory, for instance in the form of the complexity class BPP, which stands for “bounded-error polynomial time.” A language L lies in this class if there is an algorithm for deciding L that on input of some x also gets some extra random bits and outputs the correct answer (namely whether $x \in L$ or $x \notin L$ holds) with probability at least $2/3$ (taken over all possible random bits). The choice of “ $2/3$ ” is, of course, somewhat arbitrary, and any number strictly larger than $1/2$ can be used instead. To increase the likelihood of a correct answer of a BPP-algorithm, one can rerun the algorithm many times and output the majority – and when “many times” means “a polynomial number of times,” the likelihood of making an error will become exponentially small. Indeed, it is easy and practical to ensure that even though the algorithm *could* output a wrong answer, it is much more likely that the computer running the algorithm is spontaneously hit by a meteor (or consumed by dragon fire, for that matter). For this reason, “being in BPP” – and not only “being in P” – is a “common theoretician’s notion of tractability.”

Background on Spectra. The set of possible values that $\Pr_{vc}[G]$ can have for different graphs G is called the *spectrum* of $\Pr_{vc}[\cdot]$, and following [30] we will denote it as VC-PR-SPECTRUM, see Figure 4 for a visualization. Clearly, the spectrum is a subset of the interval $[0, 1]$ and it is not hard to see that many values are *not* part of this spectrum:

- The number 0 is not in the spectrum as every graph has a vertex cover (the probability that a random vertex set covers the graph is never 0).
- The number $1/3$ is not in the spectrum as $\Pr_{vc}[G]$ is always of the form $x/2^y$ for integers x and y (such numbers are called *dyadic* and the spectrum only contains dyadic numbers).

- No number strictly between $\frac{3}{4}$ and 1 is in the spectrum: Once there is a single edge in the graph, at most three quarters of all vertex sets cover this single edge (and hence, only less sets can cover the whole graph).
- Likewise, no number strictly between $\frac{5}{8}$ and $\frac{3}{4}$ can be in the spectrum, since a second edge in G already drops $\Pr_{\text{vc}}[G]$ to at most $\frac{5}{8}$.

So, we see that there are some “gaps” in the spectrum near 1, namely the empty intervals $(\frac{3}{4}, 1)$ and $(\frac{5}{8}, \frac{3}{4})$. Rephrased in terms for the following definition of the “spectral gap below p ,” we have $\text{spectral-gap}_{\text{vc}}(1) = \frac{1}{4}$ and $\text{spectral-gap}_{\text{vc}}(\frac{3}{4}) = \frac{1}{8}$:

Definition 4.2. $\text{spectral-gap}_{\text{vc}}(p) := \sup\{\epsilon \geq 0 \mid (p - \epsilon, p) \cap \text{VC-PR-SPECTRUM} = \emptyset\}$.

However, these gaps must rapidly get smaller as we get near to $\frac{1}{2}$: Each star (that is, each $K_{1,k}$) has $\Pr_{\text{vc}}[K_{1,k}] = \frac{1}{2}(1 + 2^{-k})$ as we can cover a star either by picking the central vertex or by picking all other vertices (which happens with probability 2^{-k}); so any gaps below different $p > \frac{1}{2}$ in the spectrum get smaller and smaller as p approaches $\frac{1}{2}$. For instance, $\text{spectral-gap}_{\text{vc}}(\frac{1}{2} + 2^{-1000}) \leq 2^{-1001}$.

Formally, $1/2$ is an accumulation point of **VC-PR-SPECTRUM** and it is tempting to assume that the spectrum below $1/2$ is simply a dense set as we have reached an accumulation point. However, something surprising happens: Even though *above* $1/2$ the gaps got smaller and smaller, directly *below* $1/2$ there is a sizable spectral gap once more!

Lemma 4.3. $\text{spectral-gap}_{\text{vc}}(\frac{1}{2}) \geq 2^{-10}$.

Proof. Recall that Algorithm 2 from the previous section internally precisely computed the number $\#\text{vc}(G)$ for graphs $G \in \text{MAJ-vc}$. A closer look at the algorithm reveals two things:

1. We actually precisely compute $\#\text{vc}(G)$ for any graph G with $\Pr_{\text{vc}}[G] > \frac{27}{64}$ (rather than only for the case “ $\dots \geq \frac{1}{2}$ ”) since all such graphs do not have a matching of size 3 and, hence, pass the test from line 4.
2. There can be at most nine $\beta: X \rightarrow \mathbb{B}$ that pass the contradiction test in line 11 (as G restricted to X contains a size-2 matching). So c is the sum of at most nine powers of two.

To reiterate, for any graph G for which $\Pr_{\text{vc}}[G]$ is “just below $\frac{1}{2}$ ” we have that $c = \#\text{vc}(G)$ is the sum of at most nine powers of two. But this means that $\Pr_{\text{vc}}[G] = 2^{-i_1} + 2^{-i_2} + \dots + 2^{-i_9}$ (or less summands, but let us focus on this case for a moment) and *in the binary representation of $\Pr_{\text{vc}}[G]$ there are (at most) nine 1s.*

All told, for any graph G with $\frac{27}{64} < p := \Pr_{\text{vc}}[G] < \frac{1}{2}$, we have $p = 0.0b_1b_2b_3b_4\dots_2$ where at most nine of the b_i are 1 (the first bit after “0.” must be

0 in order to have $p < \frac{1}{2}$). But the largest possible number with this property is $p = 0.011111111_2 = \frac{1}{2} - 2^{-10}$ proving the claimed lower bound on the size of the spectral gap below 1/2. \square

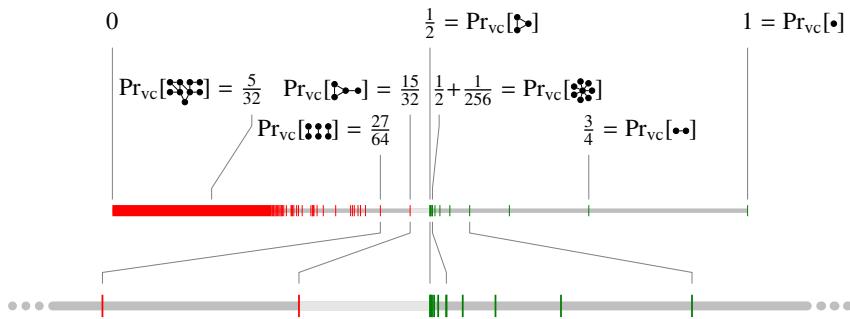


Figure 4: Visualization of vc-PR-SPECTRUM, the set of all values that $\Pr_{vc}[G]$ can have for any undirected graph G . Each line represents one possible value, with some example graphs shown that have this value as their cover probability, and red lines indicate values that are less than $\frac{1}{2}$ while green lines correspond to values at least $\frac{1}{2}$. The lower part is just a zoom where the *spectral gap* between $\frac{15}{32}$ and $\frac{1}{2}$ is easier to see. No graph has a cover probability that lies in this interval, as shown in Lemma 4.3 (for a less tight bound, though).

The bound from the lemma is not necessarily tight as it is not clear whether there actually is a graph G whose satisfaction probability is below 1/2 and is exactly the largest possible sum $\sum_{i=2}^{10} 2^{-i}$ of nine powers of two with this property. Indeed, it turns out that such a graph does not exist and we will see in the proof of Theorem 7.12 that the spectral gap is, in fact, $\text{spectral-gap}_{vc}(\frac{1}{2}) = 2^{-5} = 1/32$. So returning to Theoretical Foods Inc. we now see that the extra information volunteered by the employee, namely that the fraction of black rice grains is either at least 50% or at most 46.875%, directly corresponds to a *spectral gap in the spectrum of fractions of black grains below 1/2 of size* $3.125\% = \frac{1}{32}$ and that this is exactly the spectral gap we have below 1/2 for the fraction of vertex covers of graphs, see also Figure 4.

The natural next question is, of course, what about other values even smaller than 1/2? Does the spectrum get dense anywhere further down? It does not! But this is perhaps now less surprisingly than earlier:

Theorem 4.4. $\text{spectral-gap}_{vc}(p) > 0$ for all $p \in (0, 1]$.

Proof. In Lemma 4.3 we argued that in the binary representation of $\Pr_{vc}[G]$ there can be at most nine 1s whenever $\Pr_{vc}[G] > \frac{27}{64}$ holds. The reason was that such

a graph cannot contain a matching of size 3 (as this would lower $\Pr_{\text{vc}}[G]$ to at most $\frac{27}{64}$) and at most nine $\beta: X \rightarrow \mathbb{B}$ can contribute to the sum in equation (1). In the same way, for any $p \in (0, 1]$, there will be a constant c such that $(\frac{3}{4})^c < p/2$ holds, so any graph G with $\Pr_{\text{vc}}[G] > p/2$ does not contain a matching of size c . This means that we can write $\#\text{vc}(G)$ as a sum of at most $j := 3^{c-1}$ powers of two and, hence, the binary representation of $\Pr_{\text{vc}}[G]$ contains at most j many 1s. Consider the binary representation $0.b_1b_2b_3b_4\cdots_2$ of p such that infinitely many of the $b_i \in \{0, 1\}$ are 1s (for a number like $p = 1/3 = 0.01010101\ldots_2$ this is automatically the case, but when p is a dyadic number like $3/8$, then we have $3/8 = 0.011_2 = 0.00111111\cdots_2$ and we choose the latter representation). Now consider any graph G with $p/2 < \Pr_{\text{vc}}[G] < p$ (if there is no such graph, the spectral gap has size at least $p/2 > 0$ and we are done). Since the binary representation of $\Pr_{\text{vc}}[G]$ contains at most j many 1s, the largest number with this property that is still strictly less than p is *the number obtained from p by setting all bits after the j th 1 to 0*. Since this number is strictly smaller than p , there is a spectral gap below p as claimed. \square

The above theorem is the special case of the *Spectral Well-Ordering Theorem* from [30] for monotone 2CNF formulas. The curious term “well-ordering” comes from the fact we can state Theorem 4.4 equivalently as “*vc-PR-SPECTRUM is well-ordered by >*,” but this is really just a different way of stating that there are “spectral gaps below all p ”.

Applying Random Sampling. With all the preparations done, it is relatively straightforward to apply random sampling to solve MAJ-VC:

Algorithm 3: A random-sampling-based algorithm for deciding MAJ-VC: We sample 2,250 many subsets of V (“grains”) and count how many of them are covers (“black grains”). We claim membership in MAJ-VC whenever the number of covers is at least $2,250 \cdot \frac{31}{64} > 1,089$ many. As shown in the proof of Theorem 4.5, this claim is correct with probability at least $2/3$.

```

1 input  $G = (V, E)$ 
2  $c \leftarrow 0$ 
3 do 2,250 times:
4    $C \leftarrow$  random subset of  $V$ 
5   if  $C$  is a cover of  $G$  then
6      $c \leftarrow c + 1$ 
7
8 if  $c > 1,089$  then output “ $G \in \text{MAJ-VC}$ ” else output “ $G \notin \text{MAJ-VC}$ ”
```

Theorem 4.5. *Algorithm 3 shows $\text{MAJ-VC} \in \text{BPP}$.*

Proof. By Lemma 4.3 we know that there is a spectral gap of size at least 2^{-10} below $p = 1/2$ in the spectrum of the function $\Pr_{\text{vc}}[\cdot]$ and the more detailed analysis in the proof of Theorem 7.12 shows that the actual size is $\frac{1}{32}$. In particular, for any graph G the probability is at least $\frac{1}{2}$ or at most $\frac{15}{32}$. But, then, plugging the values $p_1 = \frac{1}{2}$ and $\epsilon = \frac{1}{64}$ into inequality (3) and and $p_2 = \frac{15}{32}$ and also $\epsilon = \frac{1}{64}$ into inequality (2) from the Chernoff–Hoeffding bound for $t = 2,250$, see Fact 4.1, we get that when we sample t random vertex sets and then check whether at least $\lceil 2,250 \cdot (p_2 + \epsilon) \rceil = \lceil 2,250 \cdot (p_1 - \epsilon) \rceil = 1,090$ of them are covers, our output will be correct with probability at least $2/3$. \square

Extensions.

- The correctness of Algorithm 3 for deciding MAJ-VC hinges on the existence of a spectral gap below $1/2$. Since we saw in Theorem 4.4 that spectral gaps “are all over the place,” the algorithm can be adjusted to $\text{MON-2SAT-PR}_{\geq p}$ for any p just by changing the number $t = 2,250$ of trials in accordance with whatever Fact 4.1 requires for a given p and $\epsilon = \text{spectral-gap}_{\text{vc}}(p)/2$ and changing the number 1,089 to $t \cdot (p - \epsilon)$. The fact that we can adapt the algorithm so easily also tells us something about the sizes of the spectral gaps: *They must get pretty small as we approach 0* since, otherwise, we could use the randomized Algorithm 3 in conjunction with binary search to compute $\Pr_{\text{vc}}[G]$ quickly. (Formally, $\#P \subseteq \text{FP}^{\text{BPP}}$ would hold.) A more detailed analysis of the sizes of spectral gaps can be found in [30].
- The one place in the algorithm that directly refers to the vertex cover problem is line 5, where we check whether our sample C is a cover. We could replace this check by any other sensible check (like “Is C a hitting set of H ?” or “Is β a satisfying assignment of ϕ ?” and so on) and we would still get a correct algorithm *as long as there is a spectral gap in the spectrum of the function for which we wish to decide a threshold*. For instance, Algorithm 3 could be used to decide, for instance, $4\text{SAT-PR}_{\geq 63/128}$, as long as $\text{spectral-gap}_{4\text{CNFS}}(63/128) > 0$ holds. By the *Spectral Well-Ordering Theorem* [30] this is, indeed, *always the case* and Algorithm 3 can be used to show $k\text{SAT-PR}_{\geq p} \in \text{BPP}$ for all $k \geq 1$ and $p \in [0, 1]$.
- On the first two ways, it was trivial to adapt the algorithm so that it also works for $\text{MON-2SAT-PR}_{>1/2}$ rather than $\text{MON-2SAT-PR}_{\geq 1/2} = \text{MAJ-VC}$. Perhaps surprisingly, trying to adapt our random sampling algorithm fails quite spectacularly. The reason is that while there was a spectral gap *below*

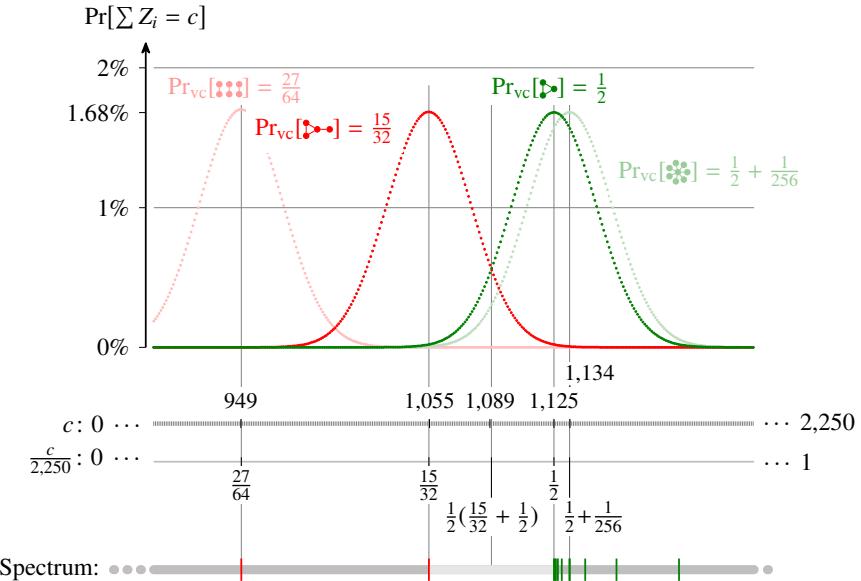


Figure 5: Plots of the probability that we see a certain count c of covers during a run of Algorithm 3: For a given graph G , the algorithm counts in c how many out of 2,250 sampled vertex sets are covers of G (for the i th sampled vertex set, the indicator variable Z_i is then 1). For the graph $\bullet\bullet$ we have $\Pr_{vc}[\bullet\bullet] = \frac{15}{32}$, so if we sample 2,250 random vertex sets, the expected number of covers (the number $\sum_{i=1}^{2,250} Z_i$) will be $\frac{15}{32} \cdot 2,250 = 1,054.6875$. The probability that on a run of the algorithm we get *exactly* 1,054 covers is $\binom{2,250}{1,054} \left(\frac{1}{2}\right)^{1,054} \left(1 - \frac{1}{2}\right)^{2,250-1,054} \approx 1.68454\%$, that we get *exactly* 1,055 covers is about 1.68501%, and that we get any $c > 1,089$ is the integral over all values to the right of the central line at 1,089. Plugging in the numbers into Fact 4.1 yields that the probability is less than 1/3 (actually, the integral is even less than 8% as the bound from the fact is not tight). For the green triangle graph, $\Pr_{vc}[\bullet\bullet] = \frac{1}{2}$, and the probability that for a random sample of 2,250 vertex sets we have $c > 1,089$ is the integral over all dark green probabilities to the right of 1,089. Applying Fact 4.1 for the appropriate values gives that the probability is at least 2/3, and it is actually even more than 93%. At the bottom, the spectrum of possible values that $\Pr_{vc}[G]$ can attain is shown (see Figure 4 for details) with a prominent *spectral gap* between $\frac{15}{32}$ and $\frac{1}{2}$. Because of this gap, for all G with $\Pr_{vc}[G] < \frac{1}{2}$ we actually have $\Pr_{vc}[G] \leq \frac{15}{32}$ (light red graphs) and $c > 1,089$ is even less likely than for $\bullet\bullet$. For graphs with $\Pr_{vc}[G] \geq \frac{1}{2}$ (light green graphs), $c > 1,089$ is even more likely than for $\bullet\bullet$.

$1/2$, there is *none above* (see Figure 4). Indeed, no matter how many samples we take, we will never be able to differentiate with high confidence between the distribution of c -values generated by a triangle (which is not an element of $\text{MON-2SAT-PR}_{>1/2}$) and the distribution generated by $K_{1,k}$ for a large k (which is an element of $\text{MON-2SAT-PR}_{>1/2}$ for all k). It turns out that this “asymmetry” between the “ $\geq p$ ” and the “ $> p$ ” cases is fundamental: Akmal and Williams [2] show, for instance, that $4\text{SAT-PR}_{>1/2}$ is NP-complete, while we just saw that $4\text{SAT-PR}_{\geq 1/2}$ lies in BPP.

5 The Fourth Way: Via Algorithmic Meta-Theorems

The employee of *Theoretical Foods Inc.* thanks you yet again for your help as you take your leave. With the silos of grain behind you, you start to explore a fourth way that quickly leads to a large patch of vegetation that looks nothing like the orderly Search Tree Nursery of your first stop, but more like a jungle: There are still lots of trees, but these are now overgrown by ivy, have lianas hanging from the branches, and you spot mistletoe between the rich leaves. A woman stands next to the winding path through the jungle, wearing some kind of uniform with the inscription *Courcelle and Partners D&C Counting Corp.* and she immediately offers their services: “Do you need help with examining any of these plants? We specialize in answering all sorts of counting problems regarding them!”

With the “plants” corresponding to input graphs, it is a good thing that they need not be trees, but rather can deviate and have additional edges. Formally, we will classify our input graphs according to how “tree-like” they are, a measure known as *tree-width*. Graphs of tree-width 1 are just trees, but already for tree-width 2 there can be all sorts of interesting out-growth and cycles have, for instance, tree-width 2. The higher the tree-width, the more complicated the graphs can be.

You politely ask what “all sorts of counting problems” encompasses and whether it includes, in particular, “counting covers.” Delightedly, she answers: “Oh, there are really *a lot* of problems we can solve very efficiently. Counting covers is actually pretty simple for us, so, yes, we can definitely help you with that.” Indeed, determining the number of vertex covers of a graph is just one of a great many of problems that can be solved using *divide-and-conquer* on graphs of fixed tree-width.

Just as you are about to commission the *Courcelle and Partners D&C Counting Corp.*, the employee mentions a caveat: “Please be aware that our services become exponentially more expensive as the tree-width increases. Is that a problem for you?” At first, this seems like *quite* a problem to you since all sorts of graphs can appear as input, even cliques with very high tree-width. But then you notice that you do not need the services for graphs of tree-width higher than four: For them,

we always have $G \notin \text{MAJ-VC}$.

In the following, we have a closer look at the background of tree-width and the services offered. Then we show how *Courcelle's Theorem* [8] can be used to decide **MAJ-VC**.

Background on Tree-Width. Suppose we wish to apply the divide-and-conquer method to solve **MAJ-VC**. It is not immediately clear how that would work on an arbitrary graph, so let us start with trees: Suppose the graph $T = (V, E)$ is a tree (a connected, acyclic, undirected graph). Pick a root $r \in V$ and let its children be $c_1, \dots, c_t \in V$, each of which is the root of a subtree $T_i = (V_i, E_i)$. Now, knowing the numbers $\#_{vc}(T_i)$ of vertex covers of each T_i , is already helpful for determining $\#_{vc}(T)$, “but not quite”: The product of these numbers is the number of vertex covers of T that include the root r , but it does not quite account for those vertex covers of T that miss r . Of course, for these, all children c_i of r must be in the cover (since all the edges from r to its children must be covered), but we do not know how many covers have these properties. The trick is to compute *two* values for each graph T in a recursion: The number of covers C of T with $r \in C$ and the number of covers with $r \notin C$. (By a similar argument as in the section on search trees, the first number is $\#_{vc}(T \setminus \{r\})$ and the second is $\#_{vc}(T \setminus N[r])$.) It is then not too hard to devise an algorithm that recursively computes $\#_{vc}(T)$ for any tree by computing these two numbers for each subtree.

Most graphs are not trees, so we can ask whether and how this idea can be taken any further. It turns out that there is a generalization of trees, namely *graphs of bounded tree-width*, for which a similar idea works: We also do a recursion on a tree (called the *decomposition tree of the graph*), but now instead of two values, we compute 2^{w+1} values for each node of the graph, where w is the tree-width of the graph (and which is obviously bounded for graphs of “bounded tree-width”).

A bit more formally, given an undirected graph $G = (V, E)$, a *tree decomposition of G* is a tree $T = (N, F)$ (whose vertices will be called *nodes* to better distinguish them from the *vertices* of G) together with a mapping *bag*: $N \rightarrow \{U \mid U \subseteq V\}$ that assigns a *bag* to each node of the tree such that the following conditions hold:

1. *Covering condition:* For each $e \in E$ there is a node $n \in N$ such that $e \subseteq \text{bag}(n)$, that is, such that the two endpoints of the edge lie in the bag.
2. *Connectedness condition:* For each vertex $v \in V$, the set $\{n \in N \mid v \in \text{bag}(n)\}$, which contains all tree nodes whose bags contain v , is connected in T .

The *width* of a tree decomposition is the maximum size of any bag minus one, that is, $\max_{n \in N} |\text{bag}(n)| - 1$. The *tree-width of a graph G* is the minimal width of any tree decomposition of G , see Figure 6 for an example.

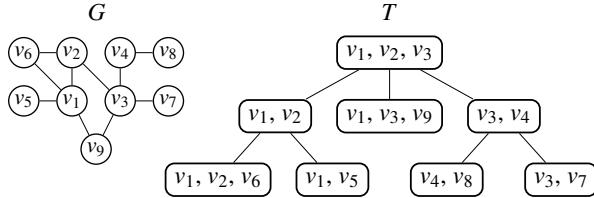


Figure 6: A graph G together with a tree decomposition T for it of width 2 (maximum bag size minus one). In the visualization, the bags attached to the nodes of T are just shown directly inside the nodes. Observe how each edge of G , such as the edge $\{v_1, v_6\}$, is a subset of some bag of T (namely the lower left bag $\{v_1, v_2, v_6\}$) and how for each vertex of G , such as v_2 , the nodes of T containing v_2 (namely those on the path from the root to the lower left node) are connected in T . The tree-width of G is 2: The depicted tree decomposition T yields an upper bound of 2 and no tree decomposition of G can have width 1 as it is easy to see that a clique of G , such as $\{v_1, v_2, v_6\}$, must be a subset of some bag of any tree decomposition.

Returning to counting covers, given a tree decomposition $T = (N, F)$ of an input graph G , we can associate with every node $n \in N$ an induced subgraph G_n of G containing all vertices ‘‘mentioned in any bag of a node below n in T ’’ (for the notion of ‘‘below n ’’ to make sense, we must pick a root of T , so let us assume that we have done this). For instance, in Figure 6 for the left child n of the root (the node whose bag is $bag(n) = \{v_1, v_2\}$), the graph G_n would encompass the nodes $\{v_1, v_2, v_5, v_6\}$ and have the form Now, we wish to compute for each node n of the tree T a vector of $2^{|bag(n)|} \leq 2^{w+1}$ numbers, namely *for each possible subset $S \subseteq bag(n)$ the number of covers C of G_n with $C \cap bag(n) = S$* . For instance, for the just-mentioned left child of the root in Figure 6 with $bag(n) = \{v_1, v_2\}$, the four numbers are how many covers of G_n there are that contain v_1 and v_2 (the number is 4), how many contain v_1 but not v_2 (only 2), how many contain v_2 but not v_1 (just 1), and how many contain neither v_1 nor v_2 (the number is 0). It is now possible to compute the 2^{w+1} numbers for any node of the tree, if one knows the numbers for all its children, meaning that we can compute the numbers recursively. This, by the way, explains why *Courcelle and Partners* charge extra for larger tree-width: The cost of keeping track of all these numbers per bag rises exponentially with the tree-width.

There are, of course, several open questions at this point: How, exactly, does the recursion work? How do we obtain a tree decomposition in the first place? And what happens when the tree-width of an input graph is, indeed, large? Fortunately, it turns out that the first two questions can be side-stepped by delegating the work to an *algorithmic meta-theorem* (although the first question is not too hard to answer

directly, see for instance [29], but invoking an algorithmic meta-theorem is way cooler). The answer to the third question was already hinted at: Graphs with large tree-width cannot be elements of MAJ-VC.

Background on Algorithmic Meta-Theorems. Let us get the problem of computing tree decompositions out of the way, first: It is not trivial and quite a bit of research has been done on that question. One culmination of this research is the following theorem, whose proof is anything but simple:

Fact 5.1 (Bodlaender's Theorem [5]). *For each tree-width w , there is a linear-time algorithm $\text{BODLAENDER}_w(\cdot)$ that maps any graph G either to a tree decomposition of G of width w or to the correct output “no width- w tree decomposition exists.”*

There are also parallel variants of this result [3] and also a logspace version [12], but the above version is more than enough for our purposes.

Let us now have another look at the divide-and-conquer approach that we sketched to count the number of vertex covers of trees (and which can be generalized to larger tree-width, but let us focus on trees for a moment): Instead of counting the number of vertex covers, we could also use the method to count the number of independent sets – but, of course, that is not very surprising as these numbers are the same. More interestingly, we can use the same idea to count the number of, say, dominating sets: Knowing for each child node how many dominating sets there are that dominate the child and how many there are that dominate everything but the child, suffices to compute these two numbers for a parent node, yielding a recursion – and this, too, generalizes to larger tree-width. Indeed, once one starts investigating this question more closely, literally *hundreds* of counting problems turn out to be amenable to applying divide-and-conquer via tree decompositions. Here is an excerpt from a paper [4] by Bodlaender from 1989:

Theorem 4.4 *Each of the following problems is in NC, when restricted to graphs with tree width $\leq K$, for constant K : vertex cover [GT1], dominating set [GT2], domatic number [GT3], chromatic number [GT4], monochromatic triangle [GT5], feedback vertex set [GT7], feedback arc set [GT8], partial feedback edge set [GT9], minimum mammal matching [GT10], partition into triangles [GT11], partition into isomorphic subgraphs for fixed H [GT12],...*

47 (!) further problems

... maximum length-bounded disjoint paths for fixed J [ND41], maximum fixed-length disjoint paths for fixed J [ND42], chordal graph completion for fixed k , chromatic index, spanning tree parity problem, distance d chromatic number for fixed d and k , thickness $\leq k$ for fixed k , membership for each class C of graphs, which is closed under minor taking.

Clearly, there must be some property that all (or at least most of) these problems share: It defies credibility that the same method just *happens* to work for all of them without some underlying reason. The honour of identifying this property is due to Bruno Courcelle, who observed that all of these problems can be *described in monadic second-order logic*.

In order not to get (further) side-tracked, no detailed account of how this logic works will be given (and it will not be important for our algorithms), a few simple examples will have to suffice (see for instance [29] for a detailed account): The idea is to interpret graphs as finite logical structures in the sense of predicate logic with the structure's universe being the vertices and interpreting the edge relation as a binary relation \sim . The subsets $C \subseteq V$ for which we wish to check whether they are covers now correspond to a set variable C . The statement “ C is a vertex cover of G ” gets translated to “(the logical structure corresponding to) G is a model of $\eta_{vc}(C) = \forall x \forall y (x \sim y \rightarrow (C(x) \vee C(y)))$ ” (we use η for formulas of predicate logic to avoid confusion with the ϕ used for propositional logic). As further examples, the statement “ C is an independent set in G ” gets translated to “ G is a model of $\eta_{is}(C) = \forall x \forall y (x \sim y \rightarrow (\neg C(x) \vee \neg C(y)))$ ”; and “ C is a dominating set of G ” gets translated to “ G is a model of $\eta_{ds}(C) = \forall x \exists y (C(y) \wedge (x = y \vee x \sim y))$.”

Fact 5.2 (Courcelle’s Theorem, Counting Version, [1]). *Let $\eta(C)$ be a monadic second-order formula with a free set variable C . Then for each tree-width w , there is a polynomial-time algorithm $\text{COURCELLE-COUNT}_{\eta,w}(\cdot, \cdot)$ that on input of any graph $G = (V, E)$ together with a width- w tree decomposition of G outputs the number of subsets $S \subseteq V$ such that $G \models \eta(S)$, that is, such that G is a model of η when S is assigned to the variable C .*

The above theorem is (one of many, many) *algorithmic meta-theorems* where the “meta” comes from the fact that for each $\eta(C)$ we get a new algorithm. Other meta-theorems differ in the logic used (some allow only less powerful logics like first-order logic, some allow more powerful ones), in the allowed class of graphs (some allow only more restricted graphs than graphs of fixed tree-width, other allow much more general graphs), and in the amount of resources used (some need only constant time, some need exponential time). It also makes a difference whether *counting* problems are considered (like above) or *decision* problems (“is there an S with $G \models \eta(S)$?”) or *construction* problems (“find an S with $G \models \eta(S)$ ”).

Applying Algorithmic Meta-Theorems.

Theorem 5.3. *Algorithm 4 shows $\text{MAJ-VC} \in \text{P}$.*

Proof. The algorithm is a straightforward application of Courcelle’s Theorem, Fact 5.2, and the only thing we have to show is that the output in line 4 is correct:

Algorithm 4: Using Courcelle's Theorem to decide MAJ-VC : We first compute a tree decomposition of width 4 of an input graph G using Bodlaender's algorithm. This may result in a failure when G actually has a larger tree-width than 4, but, then, we can output that $G \notin \text{MAJ-VC}$ holds (see the proof of Theorem 5.3 for why this is correct). Otherwise, we have a tree decomposition of G of small width and can apply a counting version of Courcelle's powerful theorem to obtain the number of vertex covers of G .

```

1 input  $G = (V, E)$ 
2
3  $T \leftarrow \text{BODLAENDER}_4(G)$  // see Fact 5.1
4 if  $T$  is “no width-4 tree decomposition exists” then output “ $G \notin \text{MAJ-VC}$ ” and stop
5
6  $c \leftarrow \text{COURCELLE-COUNT}_{\forall x \forall y (x \sim y \rightarrow (C(x) \vee C(y))), 4}(G, T)$  // see Fact 5.2
7 if  $c \geq 2^{|V|}/2$  then output “ $G \in \text{MAJ-VC}$ ” else output “ $G \notin \text{MAJ-VC}$ ”
```

We must argue that when the tree-width is larger than 4, then $G \notin \text{MAJ-VC}$ holds; or, by contraposition, that all $G \in \text{MAJ-VC}$ have tree-width at most 4. For this, let $G \in \text{MAJ-VC}$ be given. By Lemma 2.1, G contains no matching of size 3. Then G cannot contain any paths of length 6 since five edges $\{u_1, u_2\}, \{u_2, u_3\}, \{u_3, u_4\}, \{u_4, u_5\}, \{u_5, u_6\}$ for six different vertices u_i would yield $\{\{u_1, u_2\}, \{u_3, u_4\}, \{u_5, u_6\}\}$ as a matching in G . So, all paths in G have length 5 or less. By Lemma 5.4 below, the tree-width of G is at most 4. \square

Lemma 5.4. *Let G be a graph in which all paths have length at most l . Then the tree-width of G is at most $l - 1$.*

Proof. We may assume that G is connected. The tree decomposition T will be a depth-first search tree of $G = (V, E)$ starting at some arbitrary root vertex r , see Figure 7 for an example. Assign bags to the nodes $n \in V$ of T as follows: Let $\text{bag}(n)$ be the set of vertices on the path from r to n in T . We claim that this yields a tree-decomposition of width $l - 1$ of G :

1. The *covering condition* is satisfied since for any edge $\{u, v\} \in E$, in any depth-first search tree, the vertices u and v lie on a path from r to some leaf of T .
2. The *connectedness condition* is satisfied since for any vertex $v \in V$, exactly the nodes in the subtree rooted at n contain v .
3. Concerning the width, note that the depth of T (the longest path from r to any leaf) is at most l since this longest path is also a path in G . Thus, all bags have size at most l and the width is $l - 1$. \square

(The lemma actually even shows that the tree-depth of G is at most l , but this concept will only be introduced and needed on a later way.)

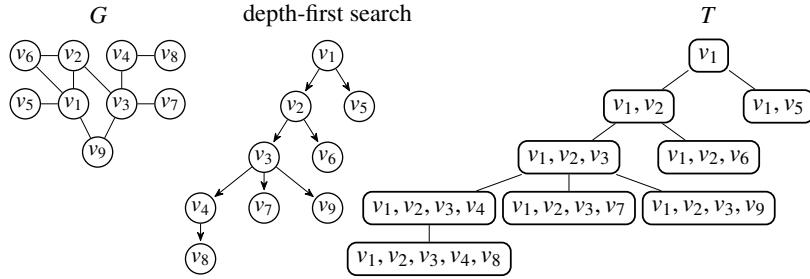


Figure 7: A graph G , the result of a depth-first search (DFS) starting at v_1 , and the resulting tree decomposition T of G where each bag for a vertex/node n contains all vertices of G that lie on the path from n to the root v_1 in the DFS tree. Note that the width of T is $4 = |\{v_1, v_2, v_3, v_4, v_8\}| - 1$ and this is “caused” by the path $v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_4 \rightarrow v_8$ in G .

Extensions.

- As for the previous ways, it is easy to generalize the presented ideas to $\text{MON-2SAT-PR}_{\geq p}$ for arbitrary p : All that needs to be changed is the bound on the tree-width in line 4. The reason is that when $\Pr_{vc}[G] \geq p$ holds, then G cannot contain large matchings (see Lemma 2.3) and hence no long path and hence cannot have large tree-width (see Lemma 5.4).
- Generalizing the approach via algorithmic meta-theorems to $2\text{SAT-PR}_{\geq p}$ and also to $2\text{SAT-PR}_{>p}$ is also pretty easy since, as hinted at, monadic second-order logic is extremely powerful (we just saw a small part of its power in the simple examples given) and it is no problem to encode negations and their behaviours into the formulas and logical structures. The arguments concerning the tree-width are not changed when negations are introduced.
- When we try to generalize our approach to $k\text{SAT-PR}_{\geq p}$ for $k \geq 3$, we run into a problem already for $\text{MAJ-3HS} = \text{MON-3SAT-PR}_{\geq 1/2}$. The trouble is not the algorithmic meta-theorem since it is quite simple to adjust the predicate logic formula $\eta(C)$ to cover hitting sets for 3-hypergraphs. The problem is that the tree-width of 3-hypergraphs (more precisely, the tree-width of the Gaifman graphs of these hypergraphs, but this is nitpicking) that are hit by at least half of all vertex sets *need no longer be bounded*: Just take any undirected

graph $G = (V, E)$ of large tree-width and form $H = (V \cup \{a\}, \{e \cup \{a\} \mid e \in E\})$ where a is a fresh vertex. Then at least half of all subsets of $V \cup \{a\}$ are hitting sets of H (namely all that contain a), but the tree-width of H is that of G plus one and, thus, large.

6 The Fifth Way: Via Graph Minors

Having politely thanked *Courcelle and Partners* for their services and having written a 5-star online review in which you praised their modest fee for tree-width 4, you continue on a fifth way. After some time you draw nearer to a building of such *gigantic* proportions that it seems to try to rival the Lonely Mountain. It is a tower that stretches all the way up into the clouds and perhaps beyond, apparently made from ivory. At the base of the tower you see colorful tents with people milling about who seem to prepare to actually *scale* the tower and, indeed, you spot climbers going up the tower on the outside, each clearly trying to reach a different location. Sometimes, climbers enter the tower through one of the many windows and then come out at the base some time later, smiling broadly. The climbers use oddly formed protrusions on the outside of the tower to help them get from one floor up to the next.

A young man approaches you and poses a somewhat puzzling question: “Hi, I am Alex. Are you also on Team Non-Planar?” Sensing your overall bewilderment, he starts explaining: “Ah, just arrived? Welcome to the Minor Tower.” You remark that the tower is anything but “minor,” which is answered by a chuckle. “Good one! This refers to the ‘minor relation,’ which allows us climbers to get from one of the protrusions (we call them graphs) on a floor to some graph on the next floor. You see those two guys there? That’s Neil and Paul and they give you a target graph and a window that you should reach. Then you start climbing (only upwards) towards that graph. The tricky part is that for each team there are certain waypoint graphs, at least one of which you have to touch first before going up to your final target. Do you see those two graphs on the fifteenth floor that looks a bit like a pentagram (❖) and like a crown (❖)?” You can, indeed, spot them. “We on Team Non-Planar have to touch them first. So, if you are not on our team, what team would you like to join? As you can see, there are many, many teams out there!”

It turns out that there is, indeed, a team than you should join (if you feel comfortable scaling towers of Babylonian proportions): Team Minority VC.

Background on Graph Minors. Given an undirected graph $G = (V, E)$, Robertson and Seymour [26] have proposed three simple ways of making the graph, well, simpler:

1. Remove an edge.
2. Remove an isolated vertex.
3. Contract an edge.

Here, contracting an edge $\{u, v\} \in E$ means removing both u and v from the graph together with all adjacent edges, but then adding a new vertex w and joining w to all former neighbours of u and v . If we start with a graph G and apply the above three operations repeatedly and end up with a graph H , we say that H is a minor of G and write $H \leq G$, Figure 8 depicts a typical example. (To be perfectly precise, we also say that H is a minor of G if some graph that is isomorphic to H is a minor of G , so we do not care about the names of the vertices.)



Figure 8: A series of modifications of a graph according to the three possible simplifications allowed by the minor relation: From the first graph to the second graph, the edge indicated in red is removed. Next, the vertex indicated in red is removed. Next, the edge indicated in red is *contracted*, that is, replaced by a vertex at the center of the former edge and all former neighbors of the endpoints are attached to the new vertex. In this way, we can simplify the graph further and further until only a single vertex remains. Any of the graphs in the sequence is a minor of all graphs to the left of it. Note that, in particular, any graph has the triangle as a minor iff it contains a cycle.

Many classes C of graphs have the property that they are “closed under taking minors,” meaning that when a graph is in the class, so is any minor of the graph:

Definition 6.1. C is closed under taking minors if $H \leq G \in C$ implies $H \in C$.

For instance, the class **PLANAR** of planar graphs is closed under taking minors: If you manage to draw a graph in a plane, removing edges or vertices clearly does not change this fact, and contracting an edge can also be done in such a way that the embedding into the plane is not destroyed. Another obvious example is the class **FORESTS** of all forests. Perhaps a bit less obvious, the classes **TREE-WIDTH- w** of graphs of tree-width at most w is closed for each w : The key insight is that if you have a width- w tree decomposition of G and apply one of the three simplifications, the tree decomposition is still valid for the simplified graph after possibly renaming some vertices in the bags. Some counterexamples are the class **MAX-DEGREE- d** of graphs of maximum degree d (as contracting an edge can raise the degree a lot) or **CONNECTED** (as removing an edge can disconnect a graph).

The great importance of the minor relation lies in the following seminal result:

Fact 6.2 (Robertson–Seymour Theorem [27], Forbidden Minor Formulation). *For every class C that is closed under taking minors, there is a finite set $\{H_1, \dots, H_s\}$ of graphs, called forbidden minors, such that: $G \in C$ iff for all $i \in \{1, \dots, s\}$ we have $H_i \not\leq G$.*

By this theorem, there must be finitely many graphs such that $G \in \text{PLANAR}$ holds iff none of these graphs is a minor of G . Such a set does, indeed, exist: By Kuratowski’s Theorem [19] it is the set $\{\bullet\text{-}\square, \bullet\text{-}\square\text{-}\square\}$. The forests are characterized by an even simpler set of forbidden minors: $\{\emptyset, \bullet\}$. In contrast, graphs of tree-width 3 are characterized [6] by the already much more complex set $\{\bullet\text{-}\square, \bullet\text{-}\square\text{-}\square, \square\text{-}\square\text{-}\square, \bullet\text{-}\square\text{-}\square\text{-}\square, \bullet\text{-}\square\text{-}\square\text{-}\square\text{-}\square\}$.

The connection to the climbers of Team Non-Planar is as follows: On the outside of the Minor Tower, the graphs are arranged in floors so that climbers can get from one graph G to a graph G' on a higher floor if $G < G'$. Then if a climber wants to reach a graph G , but must first go via $\bullet\text{-}\square$ or $\bullet\text{-}\square\text{-}\square$, they can only do so if G is non-planar.

Of course, the results on graph minors presented up to now are purely graph theoretic – we still need a link to algorithmics. This link is provided by the fact that it is possible to efficiently check for a fixed graph H whether for another graph G we have $H \leq G$:

Fact 6.3 ([17]). *For each H there is a quadratic-time algorithm for deciding on input G whether $H \leq G$.*

In the parlance of fixed-parameter tractability theory, the minor relation is fixed parameter tractable when the left relation side is the parameter. In parlance of the Minor Tower, we can decide in quadratic time whether one can climb from a fixed waypoint graph to a graph anywhere higher up (and, hence, can also find a path for the climber in polynomial time).

Joining the Robertson–Seymour Theorem and the fixed-parameter tractability of the graph minor relation, we see that we can check in polynomial time whether a graph G is, say, planar simply by testing whether $\bullet\text{-}\square$ or $\bullet\text{-}\square\text{-}\square$ is a minor of G ; and likewise for any other graph class that is closed under taking minors.

Applying Graph Minors. It should have become clear by now that we can apply the (really, really) powerful machinery of the Robertson–Seymour Theorem whenever a class of graphs is closed under taking minors. The obvious question is, then, what about MAJ-vc ? With all the effort and preparations, it is hardly surprising that we will prove in a moment that MAJ-vc is, indeed, closed under taking minors, see Corollary 6.5. Figure 9 depicts how this translates into “Team Minority VC climbing the minor tower.” The corollary follows directly from the following stronger result:

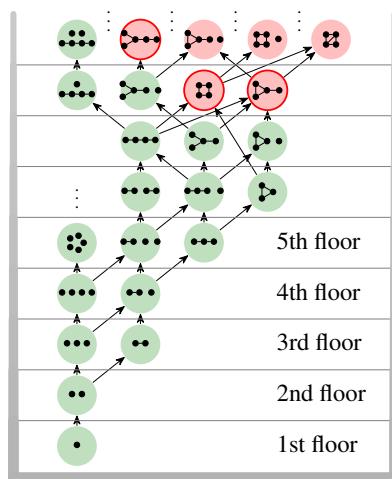


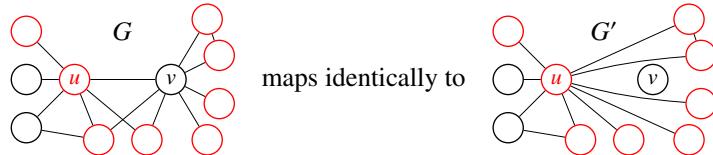
Figure 9: The Minor Tower with the first five floors shown completely (except for graphs having self-loops, to keep it simple). Climbers can get from graphs H on one floor to graphs G on the next floor (or sometimes also one higher up) if these are “in reach” as indicated by arrows. When a climber can get from H to G via several floors, $H \leq G$ holds, that is, H is a minor of G . The graphs in green are in MAJ-vc, the red ones are not. Note how MAJ-vc is “downward closed”: Once you have climbed out of it, you cannot get back into it by climbing up. Some red graphs have a red circle around them: They have the special property that *every* red graph can be reached from one of them, see Theorem 6.6.

Lemma 6.4. Let $G' \preceq G$. Then $\text{Pr}_{\text{vc}}[G'] \geq \text{Pr}_{\text{vc}}[G]$.

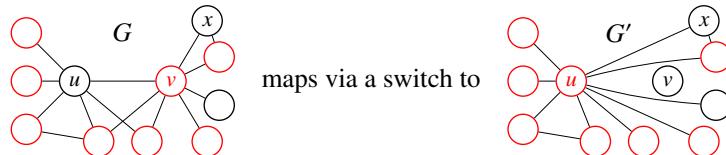
Proof. We show the claim just for the case that $G' = (V', E')$ results from $G = (V, E)$ by a single application of one of the simplification rules, the general statement then follows by induction on the number of simplifications needed (which is always finite). Clearly, removing an edge can only increase the number of vertex covers and removing an isolated vertex does not change the number at all. Thus, let G' result from G through the contraction of an edge $\{u, v\}$. Rather than adding a new vertex after removing u and v , we can also think of such a contraction as disconnecting v from all its neighbors and then making u a neighbor of this former neighborhood:



We claim that there is an injective mapping of covers C of $G = (V, E)$ to covers of $G' = (V, E')$: If C is also a cover of G' , map it to itself:



Otherwise, $u \notin C$ and $v \in C$ must hold and there must be some $x \notin C \cup \{u, v\}$ with $\{x, v\} \in E$. Map C to $C' = (C \cup \{u\}) \setminus \{v\}$.



To see that the whole mapping is injective, note that C' was *not* a cover of G since it does not cover $\{x, v\} \in E$. \square

Corollary 6.5. MAJ-vc is closed under taking minors.

Theorem 6.6. Algorithm 5 decides MAJ-vc in quadratic time.

Algorithm 5: Any graph property that is closed under taking minors, see Definition 6.1, can be decided in quadratic time with the following algorithm, provided the set of forbidden graph minors in line 3 is replaced appropriately. It is shown in the proof of Theorem 6.6 that this particular set characterizes MAJ-VC.

```

1 input  $G = (V, E)$ 
2
3 foreach  $H \in \{\text{••}, \text{•••}, \text{••••}, \text{•••••}, \text{••••••}\}$  do
4   if  $H$  is a minor of  $G$  then // use Fact 6.3 for this check
5     output “ $G \notin \text{MAJ-VC}$ ” and halt
6
7 output “ $G \in \text{MAJ-VC}$ ”
```

Proof. We know by Corollary 6.5 in conjunction with the Robertson–Seymour Theorem, Fact 6.2, that there must be *some* finite set $F = \{H_1, \dots, H_s\}$ of forbidden minors for the class MAJ-VC. Thus, if we use that set F in line 3, we get a correct decision procedure for MAJ-VC that runs in quadratic time by Fact 6.3. It remains to argue that the particular set $\{\text{••}, \text{•••}, \text{••••}, \text{•••••}, \text{••••••}\}$ is the sought set F . To see this, first note that for all graphs $H \in F$ we have $\Pr_{\text{vc}}[H] < \frac{1}{2}$ as $\Pr_{\text{vc}}[\text{••}] = \frac{1}{4}$, $\Pr_{\text{vc}}[\text{•••}] = \frac{3}{8}$, $\Pr_{\text{vc}}[\text{••••}] = \frac{15}{32}$, $\Pr_{\text{vc}}[\text{•••••}] = \frac{7}{16}$, $\Pr_{\text{vc}}[\text{••••••}] = \frac{7}{16}$, $\Pr_{\text{vc}}[\text{•••••••}] = \frac{15}{32}$, and $\Pr_{\text{vc}}[\text{••••••••}] = \frac{27}{64}$. We must now argue that for any graph G with $\Pr_{\text{vc}}[G] < \frac{1}{2}$ we have $H \leq G$ for one of the $H \in F$. For this, let us go over the possible cases:

1. Suppose G contains a loop at some vertex v . Then it must contain at least one further edge (otherwise $\Pr_{\text{vc}}[G] = \Pr_{\text{vc}}[\text{•}] = \frac{1}{2}$ would hold, contradicting the assumption $\Pr_{\text{vc}}[G] < \frac{1}{2}$). If the extra edge is the only other edge and if one endpoint is v , then $G = \text{••}$ and we would also have $\Pr_{\text{vc}}[G] = \frac{1}{2}$. Thus, the extra edge must be $\{u, w\}$ for $u \neq v$ and $w \neq v$. But then one of the two $H \in \{\text{••}, \text{•••}\} \subseteq F$ is a minor of G .
2. Suppose G contains a triangle $\{a, b, c\}$. Then it must once more contain at least one further edge. If this edge is attached to a or b or c , then $\text{•••} \leq G$. Otherwise, the edge must be separate from the triangle. Removing one edge from the triangle yields a path of length 2, showing that $\text{••••} \leq G$.
3. Suppose G contains a cycle of length 4 or longer. Then $\text{•••} \leq G$.
4. Suppose G is a forest with at least three trees (each having at least one edge). Then $\text{••••} \leq G$.
5. Suppose G is a forest with two trees (each having at least one edge). If both are just an edge, then $G = \text{•••••}$ and $\Pr_{\text{vc}}[\text{•••••}] = \frac{9}{16} \geq \frac{1}{2}$. Otherwise, $\text{••••} \leq G$.

6. Suppose G is a tree with at least one edge. If G contains a path of length at least four edges, then  $\leq G$. Otherwise, G must be a star where at most one leaf we have added an edge. If G is a star without such an added edge, $\text{Pr}_{\text{vc}}[G] > \frac{1}{2}$. If G is a star with up to two leaves and an edge added to one of them, then G is a triangle or a path of at most three edges and $\text{Pr}_{\text{vc}}[G] \geq \frac{1}{2}$. The only remaining case is that G is a star with at least three leaves, to one of which an edge is added. But, then,  $\leq G$ or  $\leq G$.

All told, whenever $\text{Pr}_{\text{vc}}[G] < \frac{1}{2}$, either G contains a cycle and, then, by the first items $H \leq G$ for some $H \in F$, or G is a forest and then, by the last items, we also have $H \leq G$ for some $H \in F$.

For future reference, observe that in all items, except for item 3, we obtain H as a minor of G purely through deleting edges and vertices, we do not need contractions (which we only need to contract long cycles in item 3). \square

Extensions.

- Using graph minors algorithmically is a very powerful method, provided the graph classes that we try to decide are closed under taking minors. We saw in Corollary 6.5 that **MAJ-vc** has this closure property since by Lemma 6.4 the minor relation is (anti)monotone with respect to covering probabilities. This implies, immediately, that for any $p \in [0, 1]$ the sets **MON-2SAT-PR** $_{\geq p}$ are also closed and, hence, have a forbidden minor characterization.
- Perhaps more intriguingly, the sets **MON-2SAT-PR** $_{>p}$ are *also* closed under taking minors and for the same reason. Thus, the graph minor method naturally also applies to these sets.
- Unlike all methods presented earlier, there is no obvious way to extend the approach to also cover **2SAT-PR** $_{\geq 1/2}$, that is, to handle negations. It is clear that we cannot just ignore the negations, but would have to come up with some sort of gadget constructions to turn formulas into graphs. However, it seems that contractions are anathema to satisfaction probabilities: Consider a set of 2CNF clauses that express that a large cycle of even length is 2-colorable. But, now, contracting (perhaps the representation of) a clause results in a formula expressing that a large cycle of odd length is 2-colorable (which is not the case). Contracting another clause yields a statement about even cycles once more, meaning that the satisfaction probabilities of the graphs representing the formulas seem to fluctuate wildly when we apply contractions.
- Another indication why it is unlikely that we can extend the method to decide, say **4SAT-PR** $_{\geq 1/2}$, is that if we could map all 4CNF formulas ϕ to graphs G_ϕ

such that $G_\phi \leq G_\psi$ implies $\Pr[\phi] \geq \Pr[\psi]$, then our approach would not only work for 4SAT-PR $_{\geq 1/2}$, but also for 4SAT-PR $_{> 1/2}$, which is known to be an NP-complete problem.

- Whether or not we can use the method to solve MAJ-3HS or, more generally, MON- k SAT-PR $_{\geq p}$, is open.

7 The Sixth Way: Via Forbidden Subgraphs

As you embark on a sixth way, you leave the mighty Minor Tower and the bustling crowd of climbers behind you. The way leads you through some beautiful theory meadows and then towards what can only be called a fairy tale castle with *a lot* of spires of different sizes. Indeed, you soon loose track of how many spires there are. At the gate of the castle, which comes with a moat and a drawbridge, a guard challenges you: “Well met, stranger. What is thy quest?” A bit unsure, you respond that you wish to solve MAJ-VC, to which the guard proclaims: “A most noble quest! Be welcome to the Castle of Thousand Spires and enter. In the Spire of Most Shallow Depths thou shalt prevail.” You thank the guard and enter.

After a bit of searching you find the Spire of Most Shallow Depths. Inside, a festive crowd of twelve dwarves makes quite a ruckus as they try to climb the spire from the inside, seldom getting very far and soon dropping back to the floor, much to the merriment of the rest. In the middle, an exasperated wizard futilely tries to impose some kind of order: “No, no, no! You must climb to $\bullet\bullet$ before you go on to $\bullet\bullet$, not the other way round!” As he sees your approach, he stops berating the dwarves and addresses you instead: “Well met, Master Theoretician!” You politely ask what is going on, to which the wizard answers: “These fine dwarves have set it upon themselves to train for the Climb of the Minor Tower, which itself is just training for climbing the Lonely Mountain. As you can see, in each spire of the Castle of Thousand Spires one can safely train climbing as getting from one graph to one on the next floor is much easier than on the Minor Tower. Once you have mastered a spire you can go to another one, they get bigger and bigger with more and more graphs in them (the ones in the Spire of Most Shallow Depths are, well, shallow). Otherwise, it is the same as with the Minor Tower: Go to the waypoints first, then to your target. Speaking of which: No, no, no, you have to touch $\bullet\bullet\bullet$, not $\bullet\bullet\bullet$!” You get a feeling that the dwarves will need to practice a *lot*, before they can ever scale the Minor Tower, let alone the Lonely Mountain. So, you casually ask why they do not use the backdoor that the hobbit showed you. Your remark is met by stunned silence, followed by a cry along the lines of “The hobbit has found the secret backdoor! It cannot be! But what, if it true? We must meet him at once!”, followed by twelve dwarves rushing out of the Spire of Most Shallow Depths. The

wizard seems deeply lost in thought, but you think you hear him mumbling: “The hobbit must have used a ring to find the backdoor. Perhaps even the nilpotent ring? I must study this!”

Let us now have a look what a “simpler way of scaling the Minor Tower” might look like and why we need many spires instead of a single tower.

Background on Induced Subgraphs. Recall from the previous section that a graph H is a minor of a graph G if we can obtain H from G by deleting edges, deleting isolated vertices, and contracting edges (and renaming vertices). If we only allow deleting edges and deleting isolated vertices, but no longer allow the contraction of edges, then H is (isomorphic to a) *subgraph* of G , written $H \subseteq G$. If we do not even allow deleting edges, but just allow deleting (no longer necessarily isolated) vertices, H is (isomorphic to an) *induced subgraph* of G , written $H \sqsubseteq G$ in the following. For isomorphic graphs H and G , we have $H \subseteq G \subseteq H \sqsubseteq G \subseteq H$. For concrete examples, $\bullet\bullet \subseteq \bullet\bullet\bullet$ and thus also $\bullet\bullet \subseteq \bullet\bullet\bullet$ and $\bullet\bullet \leq \bullet\bullet\bullet$. In contrast, $\bullet\bullet\bullet \not\subseteq \bullet\bullet\bullet$, but $\bullet\bullet\bullet \subseteq \bullet\bullet\bullet$ and thus $\bullet\bullet\bullet \leq \bullet\bullet\bullet$. Finally, $\bullet\bullet \not\subseteq \bullet\bullet\bullet$ and $\bullet\bullet \not\subseteq \bullet\bullet\bullet$, but still $\bullet\bullet \leq \bullet\bullet\bullet$.

Checking for a fixed H whether $H \sqsubseteq G$ holds, is easy enough, as the following analogue of Fact 6.3 shows:

Lemma 7.1. *For each $H = (V_H, E_H)$ there is an algorithm that on input $G = (V, E)$ decides both $H \sqsubseteq G$ and $H \subseteq G$, and that runs in time $O(|E| \cdot |V|^{|V_H|}) = |V|^{O(1)}$.*

Proof. Iterate over all size- $|V_H|$ vertex set of G and check for each whether it induces H (or a supergraph of H for \subseteq). \square

Intuitively, it feels much easier to check whether $H \sqsubseteq G$ holds than to check $H \leq G$. But Fact 6.3 tells us that the test $H \leq G$ is fixed-parameter tractable with respect to the parameter H , while testing $H \sqsubseteq G$ is easily seen to be W[1]-hard with respect to the same parameter (as setting $H = K_k$ to size- k cliques shows that the problem is at least as hard as the parameterized clique problem). So, the intuition is wrong.

Nevertheless, working with induced subgraphs or just subgraphs instead of minors has a *big* advantage: Results on the subgraph relation will generalize naturally to 2CNF and partly even to kCNF formulas. For instance, $H \subseteq G$ trivially implies $\Pr_{vc}[H] \geq \Pr_{vc}[G]$ (less edges can only mean more covers), so we have:

Lemma 7.2. *MAJ-vc is downward closed with respect to \sqsubseteq and \subseteq .*

Crucially, we likewise have that if every clause of a kCNF formula ϕ is also a clause of another CNF formula ψ (after possible variable renaming), then $\Pr[\phi] \geq \Pr[\psi]$. In other words, it is easy to come up with generalizations of the above

lemma for CNF formulas instead of graphs. In contrast, while by Lemma 6.4 we also had that $H \leq G$ implies $\text{Pr}_{\text{vc}}[H] \geq \text{Pr}_{\text{vc}}[G]$, that implication immediately breaks down when we try to add negations.

Background on Forbidden Induced Subgraphs. In order to “switch” from the minor relation to the induced subgraph relation for solving MAJ-VC, we need an analogue of the Robertson–Seymour Theorem. That is, we need a theorem telling us something like this: “Whenever a graph class C is downward closed under \sqsubseteq , then there is a finite set F of graphs such that $G \in C$ iff $H \not\sqsubseteq G$ holds for all $H \in F$. ” This statement, however, is false, as the class C of all graphs that do not contain any graph in the following set I as an induced subgraph shows (note how no graph in the infinite I is a subgraph of any other):

$$I = \{\square, \square\square\square\square, \square\square\square\square\square\square, \square\square\square\square\square\square\square, \dots\}. \quad (4)$$

Fortunately, Nešetřil and Ossona de Mendez [24] came up with a way to save the idea: We need to restrict attention only to graphs with constant *tree-depth*, as in the Spire of Most Shallow Depths, which is defined as follows:

Definition 7.3. The *tree-depth* of a graph $G = (V, E)$ is as follows: When G is the empty graph, it is 0. When G is unconnected, it is the maximum tree-depth of G ’s components. When G is connected, it is 1 plus the minimum tree-depth of $G \setminus \{v\}$ taken over all $v \in V$.

Since the definition is in terms of recursively eliminating vertices and results in an *elimination tree*, the tree-depth is sometimes also known as (*recursive*) *elimination depth*. It is closely related to concepts we saw earlier, namely *dfs trees* and *tree-width*, see Figure 10.

Let $\text{SPIRE}_t = \{G \mid G \text{ has tree-depth at most } t\}$. Figure 10 shows an example of a graph $G \in \text{SPIRE}_4$ as the depth of the depicted elimination tree is 4. Nešetřil and Ossona de Mendez [24] proved an analogue of the Robertson–Seymour Theorem for the graphs in each SPIRE_t and for the induced subgraph relation \sqsubseteq , see Theorem 7.4 below. Since we wish to generalize the underlying ideas on the seventh and final way, let us have a closer look at the proof of the theorem.

Theorem 7.4 ([24]). *Let $C \subseteq \text{SPIRE}_t$ be a graph class that is downward closed under \sqsubseteq . Then there is a finite set F of graphs such that for all $G \in \text{SPIRE}_t$ we have: $G \in C$ holds iff $H \not\sqsubseteq G$ for all $H \in F$.*

Proof. We start with a definition:

Definition 7.5. A sequence (G_0, G_1, G_2, \dots) of graphs, also written as just (G_i) , is *good*, if $G_i \sqsubseteq G_j$ holds for some indices $i, j \in \mathbb{N}$ with $i < j$; otherwise it is *bad*.

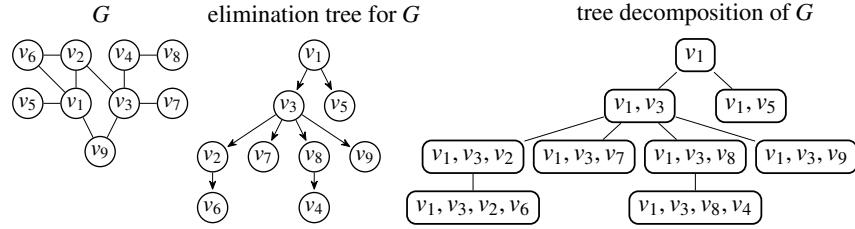


Figure 10: The graph G from Figure 7 on page 33 together with an elimination tree: When we remove the vertex v_1 , we get two connected components (one with just v_5 and the other with the remaining vertices). Then, removing v_3 from the large component leaves us with four components ($\{v_2, v_6\}$, $\{v_4, v_8\}$, $\{v_7\}$, $\{v_9\}$). Note that the depicted elimination tree is *not* a DFS tree (there is no edge between v_1 and v_3 in G), but every DFS tree like the one from Figure 7 is an elimination tree (elimination trees of a graph can be more shallow than any DFS trees for the graph). Just as for DFS trees, every elimination tree provides us with a tree decomposition of the graph by putting all vertices along the paths from the root to a node into the node's bag.

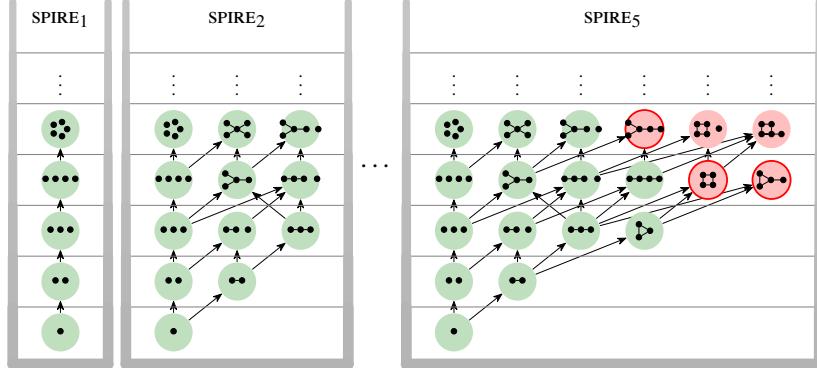


Figure 11: The three spires SPIRE₁, SPIRE₂, and SPIRE₅ depicted as in Figure 9, so green graphs are in MAJ-VC while red graphs are not, but now an arrow from H to G means that $H \sqsubseteq G$, that is, H is an induced subgraph of G (and not only a minor). In the first spire, for tree-depth 1, there are only edge-less graphs. In the second, the graphs are forests of stars. Both spires miss elements of MAJ-VC like the triangle. In SPIRE₅, we have MAJ-VC \subseteq SPIRE₅. As in Figure 9, minimal elements of the complement of MAJ-VC are indicated by a red circle (but now minimality is with respect to \sqsubseteq rather than the minor relation \preceq).

The importance of this definition lies in the following claim:

Claim 7.6. *If every sequence (G_i) with $G_i \in \text{SPIRE}_t$ is good, then there is a finite set F such that for all $G \in \text{SPIRE}_t$ we have: $G \notin C$ holds iff $H \sqsubseteq G$ for some $H \in F$.*

Proof. Let F contain all $H \in \text{SPIRE}_t \setminus C$ for which there is no $H' \sqsubseteq H$ with $H' \in \text{SPIRE}_t \setminus C$ and $H \not\sqsubseteq H'$. Consider any $G \in \text{SPIRE}_t$. Clearly, if $G \in C$, no $H \in F$ exists with $H \sqsubseteq G$ as C is downward closed. So suppose $G \notin C$. If $G \in F$, then there is an $H \in F$ (namely G) with $H \sqsubseteq G$. Otherwise, there is a $G' \sqsubseteq G$ with $G' \in \text{SPIRE}_t \setminus C$ and $G \not\sqsubseteq G'$. If $G' \in F$, then there is an $H \in F$ (namely G') with $H \sqsubseteq G$. Otherwise, there is a $G'' \sqsubseteq G$ with $G'' \in \text{SPIRE}_t \setminus C$ and $G'' \not\sqsubseteq G'$; and $G'' \in F$ once more implies that there is an $H \in F$ with $H \sqsubseteq G$. We need to repeat this argument only a finite number of times since, otherwise, $(G, G', G'', G''', \dots)$ would form a bad sequence. \square

By the claim, in order to prove the theorem, it suffices to show that every sequence in SPIRE_t is good – and this is exactly Claim 7.11 below. To prove it by induction on t , we need a slight strengthening: Let us generalize the concepts to *colored* graphs $G = (V, E, c)$, where there is a fixed set C of colors and the function $c: V \rightarrow C$ assigns a color to each vertex. A colored graph H is an induced subgraph of a colored graph G if we still can obtain H (with the correct vertex colors and with, possibly, some vertex renamings) from G by deleting vertices. We can now reach our goal through a sequence of smaller claims:

Claim 7.7. *Suppose every sequence (G_i) of connected colored graphs $G_i \in \text{SPIRE}_t$ is good. Then for every such sequence there is an infinite sequence (i_0, i_1, i_2, \dots) of strictly increasing indices such that $G_{i_0} \sqsubseteq G_{i_1} \sqsubseteq G_{i_2} \sqsubseteq \dots$.*

Proof. Suppose no such subsequence exists for (G_i) . Then starting at $G = G_0$ there must be a subsequence $G \sqsubseteq G' \sqsubseteq G'' \sqsubseteq \dots \sqsubseteq G_0^*$ of maximal length. If we remove the initial segment up to G_0^* from (G_i) , then starting at the new first graph G , there is once more a subsequence $G \sqsubseteq G' \sqsubseteq G'' \sqsubseteq \dots \sqsubseteq G_1^*$ of maximal length. By once more removing the initial segment up to G_1^* and repeating the argument indefinitely, we get an infinite subsequence (G_i^*) . But since this sequence is good, $G_i^* \sqsubseteq G_j^*$ for some $i < j$, contradicting that the sequence ending at G_i^* had maximal length. \square

Claim 7.8. *Let (G_i) be a sequence of colored graphs, where each G_i has just one vertex. Then (G_i) is good.*

Proof. After at most $|C|$ steps we see the same colored graph once more. \square

Claim 7.9. *Suppose every sequence (H_i) of connected colored graphs $H_i \in \text{SPIRE}_t$ is good. Then so is any sequence (G_i) of (possibly unconnected) colored graphs $G_i \in \text{SPIRE}_t$.*

Proof. If there are any bad sequences (G_i) of colored graphs in SPIRE_t , consider those for which G_0 has a minimal number of connected components, and then among those for which G_1 has a minimal number of connected components, then among those for which G_2 has a minimal number, and so on. No G_i can be empty (as $G_i \sqsubseteq G_{i+1}$ would follow and show that the sequence is good), so each G_i is the disjoint union of a connected H_i and some rest graph R_i , both of which lie in SPIRE_t . By assumption, (H_i) is good and by Claim 7.7 we have $H_{i_0} \sqsubseteq H_{i_1} \sqsubseteq H_{i_2} \sqsubseteq \dots$ for some indices $i_0 < i_1 < i_2 < \dots$. Consider the sequence

$$(G_0, G_1, \dots, G_{i_0-1}, R_{i_0}, R_{i_1}, R_{i_2}, \dots).$$

Since R_{i_0} has one connected component less than G_{i_0} , this sequence cannot be bad as we would have picked it at the beginning. But then either for one of the initial G_i we have $G_i \sqsubseteq R_{i_j}$ and then also $G_i \sqsubseteq G_{i_j}$ with $i < i_j$, or we have $R_{i_j} \sqsubseteq R_{i_k}$ for $i_j < i_k$ and then also $G_{i_j} \sqsubseteq G_{i_k}$. In either case, (G_i) was not bad. \square

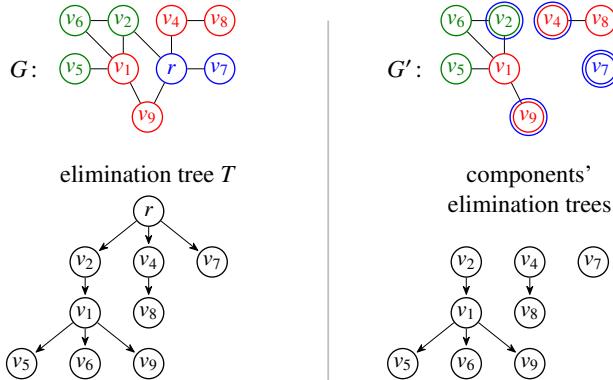


Figure 12: The graph G from Figure 10 but with a different elimination tree (which happens to also be a DFS tree, but this is not relevant). The tree's depth is 4 and hence $G \in \text{SPIRE}_4$. An example coloring for $C = \{\text{red, green, blue}\}$ is shown. If we remove r from G , we get a new graph G' with three components, each of which has an elimination tree of depth 3, showing $G' \in \text{SPIRE}_3$. In the construction of Claim 7.10, the color of r (blue in the example) is added as a second color ring to all neighbors of r , so the color of v_4 in G' is the formal pair (red, blue) and of v_7 is (blue, blue) , while the color of v_5 in G' is the pair $(\text{green, } \perp)$.

Claim 7.10. Suppose every sequence (H_i) of colored graphs $H_i \in \text{SPIRE}_t$ is good. Then so is any sequence (G_i) of connected colored graphs $G_i \in \text{SPIRE}_{t+1}$.

Proof. Let C be the set of colors and let (G_i) be a sequence of connected colored graphs $G_i \in \text{SPIRE}_{t+1}$. By definition, each G_i has an elimination tree (see Figure 10) of depth $t + 1$, rooted at some root vertex r . Build a new colored graph G'_i with the new set of colors $C \times (C \cup \{\perp\})$ as follows: Remove r from $G_i = (V_i, E_i)$ and each vertex $v \in V_i \setminus \{r\}$ with the old color $c(v)$ gets the new color $(c(v), c(r))$ if $\{v, r\} \in E_i$ and otherwise gets the color $(c(v), \perp)$, see Figure 12 for an example. In other words, we mark the vertices of G'_i by whether they were connected to r and, if so, with r 's color. The two crucial observations are that (1) if $G'_i \sqsubseteq G'_j$, then $G_i \sqsubseteq G_j$, see Figure 13 for an example; and (2) that each G'_i is an element of SPIRE_t . Then by assumption, (G'_i) is good and hence also (G_i) . \square

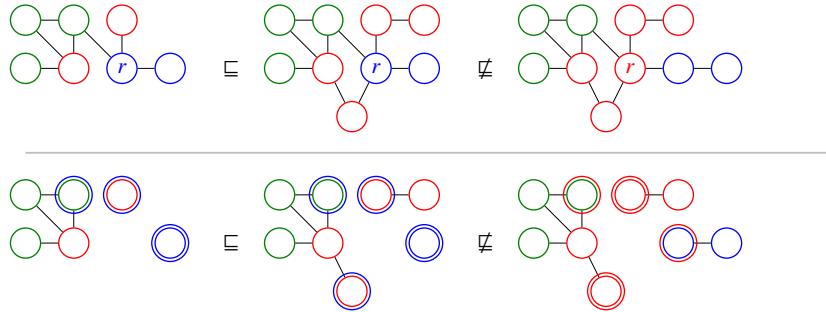


Figure 13: For the upper three graphs, the first is an induced subgraph of the second, while they are not induced subgraphs of the third as the color of r is wrong. The lower three graphs, which result from the construction from Claim 7.10 and which are in a smaller spire, reflect these induced subgraph relations.

Putting the claims together, we get:

Claim 7.11. *For each t , every sequence of colored graphs in SPIRE_t is good.*

Proof. By induction on t . For $t = 1$, Claim 7.8 gives the statement for sequences of connected graphs in SPIRE_1 and Claim 7.9 then also for unconnected graphs in SPIRE_1 . For the inductive step from t to $t + 1$, Claim 7.10 gives the statement for connected graphs in SPIRE_{t+1} and Claim 7.9 once more generalizes it to all graphs in SPIRE_{t+1} . \square

This concludes the proof of the analogue of the Robertson–Seymour Theorem for forbidden induced subgraphs and graphs of bounded tree-depth. \square

Applying Forbidden Subgraphs. Although Theorem 7.4 and the whole discussion up to now was about *induced* subgraphs, for our algorithms it is more convenient to use subgraphs. (However, we could easily reformulate the algorithms to use induced subgraphs, by adding to the set in line 6 all graphs that can be obtained by adding edges.)

Algorithm 6: The algorithm is similar Algorithm 5, but needs a new check (line 3) and the main check “if H is a minor of G ” is replaced by “if H is a subgraph of G . As shown in Theorem 7.12, the set in line 6 is the correct finite set of forbidden subgraphs to check to decide MAJ-VC.

```

1 input  $G = (V, E)$ 
2
3 if  $G \notin \text{SPIRE}_5$  then
4   output “ $G \notin \text{MAJ-VC}$ ” and halt
5
6 foreach  $H \in \{\text{•••}, \text{••••}, \text{•••••}, \text{••••••}, \text{•••••••}\}$  do
7   if  $H \subseteq G$  then // use Lemma 7.1 to check this in time  $O(n^6)$ 
8     output “ $G \notin \text{MAJ-VC}$ ” and halt
9
10 output “ $G \in \text{MAJ-VC}$ ”
```

Theorem 7.12. *Algorithm 6 decides MAJ-VC in polynomial time.*

Proof. A quick comparison reveals that there are only two deviations in Algorithm 6 from Algorithm 5: First, in line 3 we check whether $G \notin \text{SPIRE}_5$ holds (which is easy to do in time $O(n^5)$ by recursively checking all possible ways to pick elimination vertices) and, if so, claim “ $G \notin \text{MAJ-VC}$ ”. This output is correct since by the remark following Lemma 5.4, the tree-depth of all graphs in MAJ-VC is at most 5, so $\text{MAJ-VC} \subseteq \text{SPIRE}_5$. The second deviation is that the minor relation got replaced by the subgraph relation. In the proof of Theorem 6.6 we use the Robertson–Seymour Theorem to obtain a set of forbidden minors; now we use Theorem 7.4 to obtain *some* finite set F of forbidden subgraphs that we can iterate over in line 6 to correctly decide MAJ-VC. It remains to argue that the particular set $F = \{\text{•••}, \text{••••}, \text{•••••}, \text{••••••}, \text{•••••••}\}$ is once more the right one. To see this, recall from the proof of Theorem 6.6, where we did that argument already for the minor relation, that we explicitly pointed out that the argument showed that $\Pr_{\text{vc}}[G] < \frac{1}{2}$ always implies not only $H \leq G$ for some $H \in F$, but even $H \subseteq G$ for some $H \in F$; *except* for item 3 in the proof, where it was argued that if a graph G contains a cycle of length 4 or longer, then $\text{••} \leq G$. However, when G contains a cycle of length 5 or longer, then $\text{•••} \subseteq G$ and $\text{•••} \in F$. \square

Extensions.

- As was already pointed out in the introduction of this section, the whole point of switching from the minor relation to the subgraph relation in the context of counting–thresholds, was the fact that if (after renaming) the clauses of a CNF formula ϕ are a subset of the clauses of another formula ψ , then $\Pr[\phi] \geq \Pr[\psi]$. This allows one to generalize the ideas from the present section to $2\text{SAT-PR}_{\geq p}$ and $2\text{SAT-PR}_{>p}$ with (relative) ease.
- However, generalizing the method to $\text{MAJ-3HS} = \text{MON-3SAT-PR}_{\geq 1/2}$, let alone $k\text{SAT-PR}_{\geq p}$, fails almost immediately as it is easy to construct 3-hypergraphs with “many covers and very long paths” as the following set of size-3 hyperedges shows: $\{\{a, x_i, x_j\} \mid 1 \leq i < j \leq n\}$ is hit (“covered”) by any set containing a , but we can “walk n steps around the x_i .” This is, of course, not a formal proof that the method cannot work, but it is the kind of problems one quickly runs into.

8 The Seventh Way: Via Well-Quasi-Orderings

You leave the Castle of Thousand Spires and embark on a last journey. The seventh way leads you deeper and deeper into a jungle. After some time, you start to hear two male voices that seem to be having a bit of an argument: “No, this is definitely *not* an old Khmer temple!” “Just look at the collection of edifices, Junior, does that not ring a bell?” “Do not call me Junior!” “Well then, *Dr. Jones*, what is your professional archaeological opinion?” “I am not sure, but it is definitely not Khmer.” “That is the most unscientific answer I have heard in a long time, Junior. It’s that upstart faculty of yours, I have no idea how any of your colleagues ever got tenure.” “Do not call me Junior!” The bickering just continues as you draw nearer, till you are suddenly besides the two men and can see what they are discussing: It is a giant sprawl of temple-like spires and towers of all kinds and heights, all of which are richly decorated with symbols on the outside. An enormous number of ropes and vines connect many of the spires with one another, leading from symbols on one to other symbols on other spires. The older of the two gentlemen, who has a quite distinguished, if old-fashioned, demeanor, is about to address you, but the younger man, who is dressed in practical brown clothes and wears a fedora, beats him to it: “Great. *Another* archaeologist! Ah, well, perhaps *you* can shed some light on what all of that means. We have *clearly* established that this is not an old Khmer temple. But *Dr. Jones* here, and admittedly also myself, cannot make any sense of these symbols. Nor why there are so many ropes.”

You take a closer look at the symbols and then inform the gentlemen that you do not know about the ropes, but you have seen them quite recently in the Castle of

Thousand Spires on the insides of the walls there. The two men exclaim in unison “Of course! The Castle of Thousand Spires!” and the younger Dr. Jones informs you that “I must find out what these ropes are for!” and then rushes off to scale the temple spires and from time to time climbs one of the ropes to get to another spire. The older Dr. Jones just shakes his head and yells to his son: “That all looks very exciting and adventurous, Junior, but do you really *need* to shimmy along these ropes?” As we will see in the following, the ropes are, indeed, useful as they turn the isolated spires into one great *well-quasi-ordering of all graphs* that is the basis for proving not only that MAJ-vc is tractable, but also generalizations.

Background on Well-Quasi-Orderings. Algorithms 5 and 6 were almost identical: On input of a graph G , we checked for a finite set F of graphs whether for some $H \in F$ we had $H \leq G$ (Algorithm 5) or $H \sqsubseteq G$ (Algorithm 6) and, if so, knew $G \notin \text{MAJ-vc}$. The reason this worked were the Robertson–Seymour Theorem (for the minor relation \leq) and Theorem 7.4 (for the induced subgraph relation \sqsubseteq and hence also for the subgraph relation \sqsubseteq). Hardly surprisingly, these sets F do not just “happen” to exist for the relations \leq , \sqsubseteq , and \sqsubseteq . Rather, there is a deeper reason: These relations are *well-quasi-orderings* of graphs.

Well-quasi-orderings are a fundamental concept from order theory. In this theory, instead of relations between just graphs, we study pairs (S, \leq) consisting of an arbitrary sets S together with a binary relation \leq on S , meaning $\leq \subseteq S \times S$. Most readers will be familiar with, for instance, *total orderings* (S, \leq) , where \leq is transitive ($a \leq b$ and $b \leq c$ implies $a \leq c$ for all $a, b, c \in S$), reflexive ($a \leq a$ for all $a \in S$), and total (exactly one of $a \leq b$ or $b \leq a$ or $a = b$ holds for all $a, b \in S$). A more general kind are *quasi-orderings* (S, \leq) , where we just require \leq to be transitive and reflexive – so for two different elements $a, b \in S$ both $a \leq b$ and $b \leq a$ can be true (this peculiarity is emphasized by “quasi”). An example from the world of graphs is (GRAPHS, \leq) and note that for any two isomorphic graphs G and G' we have $G \leq G'$ and $G' \leq G$. Even $(\text{GRAPHS}, \sqsubseteq)$ is “just” a quasi-ordering as we defined $G \subseteq G'$ to mean that some isomorphic copy of G is a subgraph of G' .

Definition 8.1. A *well-quasi-ordering* (S, \leq) is a quasi-ordering such that for all sequences (s_0, s_1, s_2, \dots) with $s_i \in S$ for all i , there are indices $i, j \in \mathbb{N}$ with $i < j$ and $s_i \leq s_j$.

The definition is, admittedly, peculiar, but astute readers will notice that a very similar notion was introduced in the proof of Theorem 7.4 in the form of “good sequences.” That is, of course, no coincidence: That proof generalizes from the special case of “the induced subgraph relation on graphs with constant tree-depth” to all well-quasi-orderings. Indeed, very astute readers may have noticed that the proof was phrased subtly in such a way that the proofs of the following

generalizations can be copied almost verbatim from there (and, thus, will not be given in the following).

- Definition 7.5 of good sequences generalizes easily: *Let (S, \leq) be a quasi-ordering. A sequence (s_i) with $s_i \in S$ for all i is good if $s_i \leq s_j$ holds for some indices $i < j$.* With this terminology, the “well” in “well-quasi-ordering” just means that every sequence is good.
- The crucial Claim 7.6, by which a set F of forbidden graphs exists when all sequences are good, generalizes as follows:

Lemma 8.2. *Let (S, \leq) be a well-quasi-ordering and let $C \subseteq S$ be downward closed under \leq , that is, $r \leq s \in S$ implies $r \in S$. Then there is a finite set $F \subseteq S$ such that for all $s \in S$, we have $s \in C$ iff $r \not\leq s$ for all $r \in F$.*

This lemma tells us that we will *always* find sets of forbidden graphs and will *always* be able to run an analogue to Algorithms 5 and 6, when (GRAPHS, \leq) is a well-quasi-ordering.

- Claim 7.7 generalizes to the statement: *Every infinite sequence (s_i) in a well-quasi-ordering (S, \leq) has an infinite subsequence $s_{i_1} \leq s_{i_2} \leq s_{i_3} \leq \dots$* In other words, sequences in well-quasi-orderings are not only “good” in the sense “at some point, $s_{i_1} \leq s_{i_2}$ will hold,” but “perfect” in the sense that “there will be $s_{i_1} \leq s_{i_2} \leq s_{i_3} \leq \dots$ ”
- Claim 7.9, by which the general goodness of sequences of connected colored graphs extends to unconnected colored graphs, is just a special case of Higman’s Lemma (since finite unconnected graphs can be seen as finite sequences of connected graphs):

Fact 8.3 (Higman’s Lemma, [14]). *Let (S, \leq) be a well-quasi-ordering. Let S^* be the set of finite sequences of elements of S ; and for $a_1 \dots a_n \in S^*$ and $b_1 \dots b_m \in S^*$ we define $a_1 \dots a_n \leq^* b_1 \dots b_m$ if there are indices $i_1 < i_2 < \dots < i_n$ with $a_1 \leq b_{i_1}$, $a_2 \leq b_{i_2}$, ..., $a_n \leq b_{i_n}$. Then (S^*, \leq^*) is a well-quasi-ordering.*

- Finally, Claim 7.11, by which every sequence of graphs in a spire is good, translates to the following (and this is how Theorem 7.4 is actually stated in the original papers):

Fact 8.4 ([24]). *$(\text{SPIRE}_t, \sqsubseteq)$ is a well-quasi-ordering for all t .*

With all these preparations, it should come as no surprise that the Robertson–Seymour Theorem can also be rephrased very briefly:

Fact 8.5 ([27]). (GRAPHS, \leq) is a well-quasi-ordering.

Note that, in contrast, $(\text{GRAPH}, \sqsubseteq)$ is *not* a well-quasi-ordering as the bad sequence implicit in equation (4) shows. It is really just each individual $(\text{SPIRE}_t, \sqsubseteq)$ that is well-quasi-ordered, not the infinite union.

Background on Algorithmic Aspects of Well-Quasi-Orderings. As stated earlier, the theory of well-quasi-orderings is purely order-theoretic and not concerned with questions of tractability. However, the existence of finite sets of forbidden elements for closed sets gives us analogues of Algorithms 5 and 6. The following theorem turns this into a formal statement, where P is the class of problems solvable in polynomial time and XP is the class of relations \leq such that for each fixed $r \in S$ there is a polynomial-time algorithm for deciding on input $s \in S$ whether $r \leq s$ holds (in FPT parlance, the left-hand side of the relation is the parameter):

Algorithm 7: Abstract version of Algorithms 5 and 6 for well-quasi-orderings (S, \leq) with $S \in P$ and $\leq \in XP$ and a set $C \subseteq S$ that is downward closed with respect to \leq . The set F is a set of forbidden elements, see Lemma 8.2, with respect to (S, \leq) and C .

```

1 input  $s$ 
2
3 if  $s \notin S$  then
4     output “ $s \notin C$ ” and halt
5
6 foreach  $r \in F$  do
7     if  $r \leq s$  then
8         output “ $s \notin C$ ” and halt
9
10 output “ $s \in C$ ”
```

Theorem 8.6. Let (S, \leq) be a well-quasi-ordering with $S \in P$ and $\leq \in XP$. Let $C \subseteq S$ be downward closed with respect to \leq . Then $C \in P$.

Proof. By Lemma 8.2, there is a finite set $F \subseteq S$ such that for each $s \in S$ we can decide whether we also have $s \in C$ by checking whether for all $r \in F$ we have $r \not\leq s$. Consider Algorithm 7 for this particular F . On input s , we first check whether $s \in S$ holds in line 3 (which can be done in polynomial time because of the assumption $S \in P$) and, if not, know that $s \notin C \subseteq S$. Otherwise we check whether any $r \in F$ has the property $r \leq s$ (which can also be done in polynomial time because of the assumption $\leq \in XP$) and, if so, again know $s \notin C$. If s passes all these tests, we correctly assert that $s \in C$ must hold. \square

A classical way of instantiating Theorem 8.6 is for the well-quasi-ordering (GRAPHS, \leq) and $C = \text{PLANAR}$: $S = \text{GRAPHS} \in P$ holds trivially, $\leq \in \text{FPT} \subseteq \text{XP}$ follows from Fact 6.3, and the class of planar graphs is clearly closed under taking minors. Thus, $\text{PLANAR} \in P$. For our purposes, we of course use (GRAPHS, \leq) and $C = \text{MAJ-VC}$, which is closed under taking minors by Corollary 6.5, to get $\text{MAJ-VC} \in P$ in the form of Theorem 6.6. Alternatively, we can instantiate Theorem 8.6 for $(\text{SPIRE}_5, \subseteq)$ and $C = \text{MAJ-VC}$ to get Theorem 7.12.

Applying Well-Quasi-Orderings. While it is certainly elegant to restate our previous results in the abstract framework of well-quasi-orderings, does this actually give us any new insights? In other words, a weary traveller might sigh: *Why did I bother to walk this way?*

To answer this, let us rephrase the discussions of possible extensions at the end of the fifth and sixth way in terms of well-quasi-orderings: (GRAPHS, \leq) is a well-quasi-ordering for which MAJ-VC is downward closed, but when we try to go from graphs to 2CNF, the set $2\text{SAT-PR}_{\geq 1/2}$ does not seem closed under any sensible version of the minor relation. In contrast, the (induced or not) subgraph relation is more robust and allows generalizations, but neither $(\text{GRAPHS}, \sqsubseteq)$ nor $(\text{GRAPHS}, \sqsubseteq)$ are well-quasi-orderings. Rather, only $(\text{SPIRE}_t, \subseteq)$ was. What we really need is a *single “robust” well-quasi-ordering on all graphs that is anti-monotone with respect to $\text{Pr}_{\text{VC}}[\cdot]$* . Fortunately, this can be achieved by adding ropes (an idea borrowed from Rado [25]):

Definition 8.7. For each constant c , define a relation \sqsubseteq^c as follows: Let $H \sqsubseteq^c G$ hold if

1. $H \sqsubseteq G$, that is, H is an induced subgraph of G , or
2. G contains a matching of size $c \cdot n_H$ where n_H is the number of vertices in H .

The idea behind this definition is that in addition to the induced subgraph relation, we add new ways to get from a graph H to a graph G (the “ropes”) that do not respect the induced subgraph relation at all. Nevertheless, the resulting relation has all the desirable properties we need:

Lemma 8.8. *For each c , $(\text{GRAPHS}, \sqsubseteq^c)$ is a well-quasi-ordering.*

Proof. Consider any sequence (G_i) of graphs. If there is a t such that $G_i \in \text{SPIRE}_t$ holds for all i , we know that $G_i \sqsubseteq G_j$ holds for some $i < j$ by Fact 8.4 and hence also $G_i \sqsubseteq^c G_j$. Otherwise, the depth of the elimination trees of the G_i grows beyond all bounds; so some G_j with $j > 0$ has an elimination tree of depth $2c \cdot |V_0|$ where $G_0 = (V_0, E_0)$. Then this G_j contains a path of length $2c \cdot |V_0|$ (recall the remark

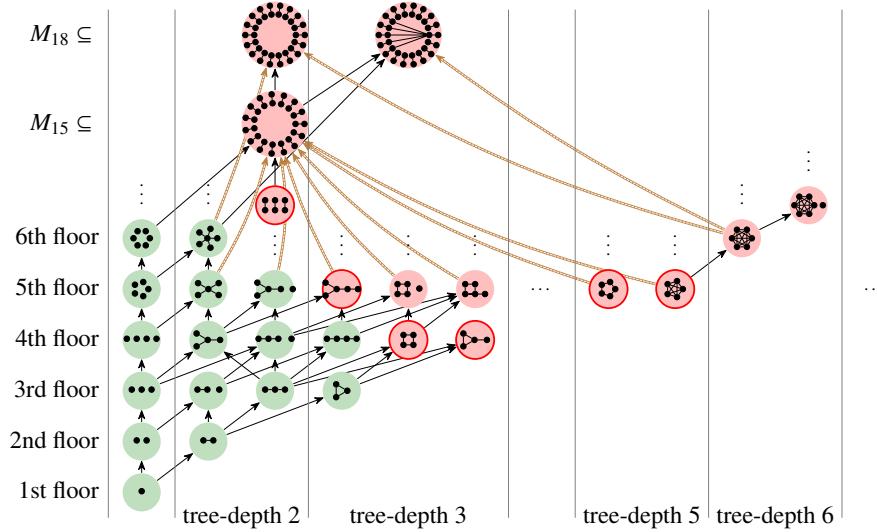


Figure 14: The well-quasi-ordering (GRAPHS, \sqsubseteq^3). As in Figure 13, the graphs are grouped according to tree-depth, but while in the spires of Figure 13 the graphs from a smaller spire got repeated, each graph is now present only once and we only use the tree-depth for visual grouping. Also as in that figure, green graphs lie in MAJ-VC while red graphs do not, and the circled red graphs are minimal elements of the complement of MAJ-VC with respect to \sqsubseteq^3 . A normal arrow (\rightarrow) still just indicates that $H \sqsubseteq G$ holds (H is an induced subgraph of G). The new “ropes” ($\xrightarrow{\dots}$) indicate that $H \sqsubseteq^3 G$ holds (only) by item 2 in Definition 8.7: The number of vertices in H (5 on the fifth floor and 6 on the sixth floor) is at most a third of the size of a matching in G (15 for graphs containing the 15-edge matching M_{15} and 18 for those containing M_{18}). Note that these ropes do not respect the subgraph relation at all, but they *do* respect $\text{Pr}_{\text{vc}}[\cdot]$ by Lemma 8.10.

after Lemma 5.4) and, hence, also a *matching* of size $c \cdot |V_0|$. Thus, $G_i \sqsubseteq^c G_j$ for $0 = i < j$. \square

Lemma 8.9. *For each c , we have $\sqsubseteq^c \in \text{XP}$.*

Proof. Fix a graph H and let n_H be the number of its vertices. To decide on input H whether $H \sqsubseteq^c G$ holds, we have to test whether either $H \sqsubseteq G$ holds, which can be done in polynomial time by Lemma 7.1, or whether G contains a matching of size $c \cdot n_H$, which can also easily be checked in polynomial time (note that n_H is a constant parameter). \square

Lemma 8.10. *$H \sqsubseteq^3 G$ implies $\text{Pr}_{\text{vc}}[H] \geq \text{Pr}_{\text{vc}}[G]$.*

Proof. If $H \sqsubseteq G$, then $\text{Pr}_{\text{vc}}[H] \geq \text{Pr}_{\text{vc}}[G]$. So assume that G contains a matching M of size $3n_H$ where n_H is the number of vertices of H . Since H has at least one cover, we have $\text{Pr}_{\text{vc}}[H] \geq 2^{-n_H}$. On the other hand, $\text{Pr}_{\text{vc}}[M]$, the probability of covering a size $c \cdot n_H$ matching, is $(\frac{3}{4})^{3n_H} = (\frac{27}{64})^{n_H} < 2^{-n_H}$. Thus $\text{Pr}_{\text{vc}}[H] \geq 2^{-n_H} > \text{Pr}_{\text{vc}}[M] \geq \text{Pr}_{\text{vc}}[G]$. \square

In particular, **MAJ-vc** is downward closed with respect to \sqsubseteq^3 . Putting it all together, we get:

Theorem 8.11. *Algorithm 7 decides **MAJ-vc** in polynomial time when instantiated for $(\text{GRAPHS}, \sqsubseteq^3)$ and $C = \text{MAJ-vc}$.*

Extensions.

- Just as not only **MAJ-vc** is closed with respect to taking minors, but all **MON-2SAT-PR $_{\geq p}$** and even all **MON-2SAT-PR $_{>p}$** are for any $p \in [0, 1]$, so are they all with respect to \sqsubseteq^3 . In other words, like the minor relation, we get the tractability of **MON-2SAT-PR $_{\geq p}$** and **MON-2SAT-PR $_{>p}$** “alongside” that of **MAJ-vc** “for free.”
- Unlike the minor relation, but like the induced subgraph relation, it is not hard (but also not *quite* trivial) to extend the definition to 2CNF formulas with negations. This allows us to give well-quasi-ordering-based proofs of $2\text{SAT-PR}_{\geq p} \in \text{P}$ and $2\text{SAT-PR}_{>p} \in \text{P}$.
- Unlike the induced subgraph relation, one *can* extend the idea to 3CNF formulas and beyond (the details are beyond the already too-large scope of this paper and will be discussed at some future time). That is, for each k it is possible to define a well-quasi-ordering \leq of all k CNF formulas such that $\phi \leq \psi$ implies $\text{Pr}[\phi] \geq \text{Pr}[\psi]$, meaning that sets such as $4\text{SAT-PR}_{>1/2}$ are closed with respect to this ordering. However, as $4\text{SAT-PR}_{>1/2}$ is known to be NP-complete, we cannot hope to have $\leq \in \text{XP}$, but only $\leq \in \text{XNP}$.

9 Conclusion

It has been quite a long journey through the complexity landscape – just to visit the problem **MAJ-VC**, which is neither particularly difficult nor particularly relevant to solve in practice. Hopefully however, the journey was the reward, with some new (in)sights gained along the way. At many vantage points we glimpsed in the distance how the presented ideas applied to more general problems like $2\text{SAT-PR}_{\geq p}$, **MAJ-3HS**, or $k\text{SAT-PR}_{\geq p}$. Experienced travellers of the complexity landscape may even have spotted applications to *constraint satisfaction problems* and *homomorphism counting* near the horizon, which are even more general than satisfiability problems, but to which many of the presented ideas still apply.

Readers who are still not yet tired may wonder how many ways of showing the tractability of **MAJ-VC** there are in total. On the one hand, one could argue that the last three ways presented in this paper are actually just *one* way as they are all variations of using well-quasi-orderings to solve **MAJ-VC**; just at different levels of abstraction. So, perhaps this paper only really presented five ways. On the other hand, there is at least one more way, presented in detail in [30], where *kernels* are used to solve $k\text{SAT-PR}_{\geq p}$. For the special case of **MAJ-VC**, this amounts to “as long as there is a star of size six, remove one ray.” However, the correctness of this idea hinges on the Spectral Well-Ordering Theorem [30] just as that of the random sampling method from the third way, so perhaps this is not another way after all.

Conversely, it is also interesting to see which ways do *not* seem to lead to solving **MAJ-VC** efficiently. For instance, it is not clear how *linear programming* could be used to decide **MAJ-VC** naturally – despite the fact that linear programming is one of the most powerful tools we have to prove the tractability of problems.

In the end, many, but not all paths lead to the tractability of **MAJ-VC** and readers are very much invited to discover new ways and to share them with their fellow travellers in the theory landscape.

References

- [1] Stefan Arnborg, Jens Lagergren, and Detlef Seese. Easy problems for tree-decomposable graphs. *Journal of Algorithms*, 12(2):308–340, 1991.
- [2] Shyan Akmal and Ryan Williams. **MAJORITY-3SAT** (and related problems) in polynomial time. In *Proceedings of the 62nd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2021)*, pages 1033–1043. IEEE Press, 2022.
- [3] Hans L. Bodlaender and Torben Hagerup. Parallel algorithms with optimal speedup for bounded treewidth. *SIAM Journal on Computing*, 27(6):1725–1746, 1998.

- [4] Hans L. Bodlaender. NC-algorithms for graphs with small treewidth. In *Proceedings of the 14th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 1988)*, volume 344 of *Lecture Notes in Computer Science*, pages 1–10. Springer, 1989.
- [5] Hans L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25(6):1305–1317, 1996.
- [6] Hans L. Bodlaender. A partial k -arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209(1):1–45, 1998.
- [7] Stephen A. Cook. The complexity of theorem-proving procedures. In *Conference Record of the Third Annual ACM Symposium on Theory of Computing (STOC 1971)*, pages 151–158, Shaker Heights, Ohio, 1971.
- [8] Bruno Courcelle. Graph rewriting: An algebraic and logic approach. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*, pages 193–242. Elsevier and MIT Press, 1990.
- [9] Rod G. Downey and Michael R. Fellows. Fixed-parameter tractability and completeness II: On completeness for W[1]. *Theoretical Computer Science*, 141(1):109–131, 1995.
- [10] Rod G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Springer, 2013.
- [11] Jan Dreier, Sebastian Ordyniak, and Stefan Szeider. SAT backdoors: Depth beats size. *Journal of Computer and System Sciences*, 142(103520):1–22, 2024.
- [12] Michael Elberfeld, Andreas Jakoby, and Till Tantau. Logspace versions of the theorems of Bodlaender and Courcelle. In *Proceedings of the IEEE 51st Annual Symposium on Foundations of Computer Science (FOCS 2010)*, pages 143–152, 2010.
- [13] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Springer, 2006.
- [14] Graham Higman. Ordering by divisibility in abstract algebras. *Proceedings of the London Mathematical Society*, s3-2(1):326–336, 1952.
- [15] David G. Harris and N. Sundararajan Narayanaswamy. A faster algorithm for vertex cover parameterized by solution size. In Olaf Beyersdorff, Mamadou M. Kanté, Orna Kupferman, and Daniel Lokshtanov, editors, *41st International Symposium on Theoretical Aspects of Computer Science (STACS 2024)*, volume 289 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 40:1–40:18, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.
- [16] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

- [17] Ken-ichi Kawarabayashi, Yusuke Kobayashi, and Bruce Reed. The disjoint paths problem in quadratic time. *Journal of Combinatorial Theory, Series B*, 102(2):424–435, 2012.
- [18] Joseph B. Kruskal. The theory of well-quasi-ordering: A frequently discovered concept. *Journal of Combinatorial Theory, Series A*, 13(3):297–305, 1972.
- [19] Kazimierz Kuratowski. Sur le problème des courbes gauches en topologie. *Fundamenta Mathematicae*, 15:271–283, 1930. In French.
- [20] Leonid Levin. Универсальные задачи перебора (Universal search problems). *Проблемы передачи информации (Problems of Information Transmission)*, 9(3):115–116, 1973. See [32, pages 399–400] for a translation to English.
- [21] Daniel Lokshtanov, Fahad Panolan, and M. Sridharan Ramanujan. Backdoor sets on nowhere dense SAT. In Mikołaj Bojańczyk, Emanuela Merelli, and David P. Woodruff, editors, *Proceedings of the 49th International Colloquium on Automata, Languages, and Programming (ICALP 2022)*, volume 229 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 91:1–91:20, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.
- [22] Michel Minoux. LRUR: a simplified linear-time unit resolution algorithm for Horn formulae and computer implementation. *Information Processing Letters*, 29(1):1–12, 1988.
- [23] Robin A. Moser and Dominik Scheder. A full derandomization of Schöning’s k -SAT algorithm. In *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing (STOC 2011)*, pages 245–252, New York, NY, USA, 2011. Association for Computing Machinery.
- [24] Jaroslav Nešetřil and Patrice Ossona de Mendez. *Sparcity: Graphs, Structures, and Algorithms*, volume 28 of *Algorithms and Combinatorics*, chapter 6: Bounded height trees and tree-depth, pages 115–144. Springer, 2012.
- [25] Richard Rado. Partial well-ordering of sets of vectors. *Mathematika*, 1(2):89–95, 1954.
- [26] Neil Robertson and Paul D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *Journal of Algorithms*, 7(3):309–322, September 1986.
- [27] Neil Robertson and Paul D. Seymour. Graph minors. XX. Wagner’s conjecture. *Journal of Combinatorial Theory, Series B*, 92(2):325–357, November 2004.
- [28] Uwe Schöning. A probabilistic algorithm for k -SAT and constraint satisfaction problems. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science (FOCS 1999)*, pages 410–414, USA, 1999. IEEE Computer Society.
- [29] Till Tantau. A gentle introduction to applications of algorithmic metatheorems for space and circuit classes. *Algorithms*, 9(3), 2016.

- [30] Till Tantau. On the satisfaction probability of k -CNF formulas. In Shachar Lovett, editor, *Proceedings of the 37th Computational Complexity Conference (CCC 2022)*, volume 234 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 2:1–2:27, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.
- [31] Seinosuke Toda. PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 20(5):865–877, 1991.
- [32] Boris A. Trakhtenbrot. A survey of Russian approaches to *perebor* (brute-force searches) algorithms. *Annals of the History of Computing*, 6(4):384–400, 1984.
- [33] Leslie G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979.
- [34] Ryan Williams, Carla P. Gomes, and Bart Selman. Backdoors to typical case complexity. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI 2003)*, pages 1173–1178, San Francisco, CA, USA, 2003. Morgan Kaufmann Publishers Inc.
- [35] David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. In *Proceedings of the Thirty-Eighth Annual ACM Symposium on Theory of Computing (STOC 2006)*, pages 681–690, New York, NY, USA, 2006. Association for Computing Machinery.

The Bulletin of the EATCS

THE EDUCATION COLUMN

BY

DENNIS KOMM AND THOMAS ZEUME

ETH Zurich, Switzerland and Ruhr-University Bochum, Germany

dennis.komm@inf.ethz.ch and thomas.zeume@rub.de

TEACHING DESIGN OF ALGORITHMS IN HIGH SCHOOLS BY CONSTRUCTIVE INDUCTION

Juraj Hromkovič

Department of Computer Science, ETH Zürich

juraj.hromkovic@inf.ethz.ch

Regula Lacher

Department of Computer Science, ETH Zürich

regula.lacher@inf.ethz.ch

Abstract

The design of algorithms is one of the hardest topics of high school computer science. This is mainly due to the universality of algorithms as solution methods that guarantee the calculation of a correct solution for potentially infinitely many instances of an algorithmic problem. The goal of this paper is to present a comprehensible and robust algorithms design strategy called “constructive induction” that enables high school students to solve a large variety of algorithmic problems. In general, this strengthens learners in problem solving and their ability to use and develop abstract representations.

1 Introduction or Why Teaching Algorithms in High Schools is Not Easy

“Why to teach algorithms?” Because it forces learners to strengthen their ability to abstract and to solve problems. Abstraction and problem solving are crucial dimensions of the human way of thinking and basic instruments for discovering and shaping the world. Therefore, abilities to abstract and solve problems should be a key issue in any educational system. Especially, the main focus of teaching mathematics and computer science has to be devoted to supporting the learner’s ability to abstract and solve problems in their abstract representation.

“Why is teaching algorithms difficult?” To answer this question let us start with relating it to the question “Why is teaching the concept of variables in introductory programming courses the first big threshold?” Good programming courses for

novices pay attention to the cognitive load of students and make sure that progress is made in very small steps. The starting point is to view programs as descriptions of activities in the programming language that are understandable for machines (computers, robots, etc.) with the goal to delegate the execution of these activities to technology. In this simplified scenario one program describes exactly one activity. If you take a good choice of your programming language and initial exercises, the first programs describe activities whose execution can be observed visually instruction by instruction (see the concept of the *notional machine* [2,4,6,8,10–12]). With this approach students can develop a program and immediately investigate its properties and functionality, and in this way verify the correctness of their program. This is the reason why even very small kids in primary school can master some basic programming with success and joy. This is also why the repeat-loop with the hidden control variable was introduced to Logo [13].

If you introduce variables, even as passive input parameters, the game changes heavily. Why is programming with variables much harder? One program is not responsible for only one activity anymore, but depending on the values of its parameters for potentially infinitely many different activities [1]. How now to check the functionality of your program? You cannot teach complete induction for proving correctness of programs or any sophisticated verification method of software engineering. For sure you can try to test your program for a few values of its parameters. Consequently, you can fail to believe in the correctness of an incorrect program, but the main problem is whether these few tests are really sufficient to get an intuition of why your program should work. And the minimal goal of any computer science teacher should be to offer at least some intuition of the program functionality.

An algorithmic problem consists of infinitely many concrete problem instances, and an algorithm is a solution method that works correctly and successfully for any one of these infinitely many instances. Hence you are asked to develop a strategy that behaves well in all infinitely many possible situations. This is a very nontrivial task, especially if you take into account that high school students have almost no experience with the universal quantifier. We cannot assume any experience with using mathematical induction for proving infinitely many parameterized claims, and so we cannot strive to prove the formal correctness of the algorithms developed. Of course we could omit verification proofs. But if our lessons have an educational value, then they have to offer some reasonable intuition, why the algorithm designed works properly. To reach this goal, teaching must be designed in such a way that students have enough freedom to design algorithms to a high extent on their own. This is a very nontrivial task we will try to approach in the subsequent sections.

Let us consider another dimension in answering our question “Why is teaching algorithms in high school not easy?” A teacher can aim to explain some famous algorithm and the goal could be to be able to execute the algorithm by hand. We

question this goal. First of all, the value of teaching acting according to a given pattern is very low [9], because it contributes very little to the development of our thinking (creativity, improvisation, fantasy). Every procedure we can describe can be automated and so executed more reliably and quicker by technology than by humans. Secondly, most efficient algorithms are so specifically adjusted to the problems they solve, that a small change in the specification of the algorithmic problem can cause the algorithm to fail. We say that such algorithms are not robust and so their teaching does not guarantee an essential progress in developing and strengthening the ability of students to solve problems. We argue that we have to teach robust strategies for problem solving that can be successfully applied to a big variety of problems. The remaining question now is which of the algorithm design methods can be taught successfully in high schools. In the following we present one of these methods, called *constructive induction*, and show how a teacher can use it in schools to train problem solving and algorithm design.

2 Constructive Induction

Everybody knows *mathematical induction* (also called *complete induction*) used to prove infinitely (countably) many parameterized claims. The word induction has its origin in Latin (“*inductio*” – “to lead into” in English). Probably the oldest induction proof was done by al-Karaji in the tenth century for proving the binomial theorem (Pascal’s triangle) about the form of coefficients [3, 5]. Unfortunately, this manuscript was lost and we only have the reference in the book “The Brilliant in Algebra” by Al Samawal al-Maghribi (around 1150). In European culture induction as a proof method was used for the first time by Francesco Maurolico in 1575 [14] for proving that the sum of the first n odd integers is n^2 . It took several years until the method was used again (Blaise Pascal, 1654; Jacob I Bernoulli, 1686). In 1888, Richard Dedekind started to call this proof method “complete induction.” Giuseppe Peano presented induction as a part of his axiomatic system in 1889. Since then, this method belongs to the fundamental instruments of mathematics. Because reading, correcting, and writing proofs is not a mandatory subject of mathematics in high school in most countries, one can question whether it is reasonable to do it in computer science lessons. But we want to deal with constructive induction here, which is easier available to students.

Constructive induction for building sequences of objects or for solving problems is much older than complete induction as a proof method and was used already in ancient time. The simplest fundamental example is the construction of natural numbers. For each number n we can construct a larger number, for instance $n+1$. In this case we use the constructive induction to construct an infinite sequence of objects. Another antic example is the claim that there are infinitely many primes.

Take the n smallest primes p_1, p_2, \dots, p_n .

Now we develop a strategy for finding the next prime p_{n+1} . Take the number

$$m = p_1 \cdot p_2 \cdot \dots \cdot p_n + 1 ,$$

which is not divisible by any of the first n primes p_1, \dots, p_n . So there must be a prime in the interval between p_n and m . Now one has to check these finitely many candidates from the smallest one to the largest one to find the next prime. What we see in this example? For any n , we have a strategy for how to find the $(n + 1)$ -th prime if you have the first n primes, and you proved that this strategy works (i.e., that there are infinitely many primes).

In the examples above we saw how we can generate step by step the next object of an infinite sequence of objects if we know the previous ones. For sure this method works only if we have or can construct the starting object of this sequence.

We can extend the constructive induction presented above for solving problems and designing algorithms. The idea is as follows. First, we have to parameterize the problem. This means that we have to partition its infinitely many instances into infinitely many classes numbered by natural numbers. The classes can be finite or infinite. For instance, a parameter of a graph can be the number of its vertices, the number of its edges, the size of its adjacency matrix, the maximum degree, or the diameter. A parameter of an integer can be the integer itself or the length of its representation in some number system. A parameter for sorting or searching can be the number of elements or the length of the whole input representation. After parameterizing a problem, we say that a problem instance is of size k if it belongs to the k -th class.

The idea of using constructive induction to solve problems is now very similar to complete induction. First, one solves the problem for the smallest parameter, i.e., the smallest problem instances. We speak about the base of the induction. Then, for any positive integer n , one executes a general strategy, how to use solutions for instances smaller than n (with parameters smaller than n), in most cases even only with parameter $n - 1$, to construct a solution for any instance of size n . The key issue here is searching for this general strategy.

In [Sections 3](#) and [4](#) we show that there are plenty of problems for which such strategies are quite natural and can be discovered successfully. This is important, because after experiencing some examples the students can start searching for algorithmic solutions to similar problems on their own. In [Section 3](#) we show how to use induction to solve problems, and in [Section 4](#) we apply constructive induction to teach high school students to design efficient algorithms.

3 Solving Problems by Constructive Induction

We present a sequence of problems that can be transparently solved by constructive induction. A detailed description of the lessons for high school students can be found in a recent textbook [7].

The number of decimal numbers with at most n digits. The question is how many decimal numbers of length at most n exist. The length of a number is the number of digits in its representation, which is the parameter of this problem. This is a very simple introductory task.

The base of the induction is easy. We have 10 digits 0, 1, 2, ..., 9, and so we have 10 integers of length 1. To discover the general step, we always encourage students to go from size 1 to size 2, from size 2 to size 3, etc. until they recognize a pattern in these concrete steps. Going from size 1 to size 2 here means to build a table of size 10×10 . In the rows there are 10 digits 0, 1, 2, ..., 9, and the columns contain the 10 decimal numbers of length 1. Combining the labels of the rows with the labels of the columns we get 100 sequences of 2 digits. Removing the leading zeros, we have all 100 decimal numbers of length at most 2. The generalization of this strategy is transparent. For the representation length at most n we take a $10 \times 10^{n-1}$ table. In the rows we have all 10 digits, and, in the columns, we have all sequences of $n - 1$ digits.

Note that this problem can be solved easier by asking which is the largest integer of length n . The answer is the number consisting of digits 9 only. But the advantage of using this task is to get a very simple introduction to constructive induction.

Lines in two-dimensional Euclidean space. Suppose we have n different lines in two-dimensional Euclidean space, and we are asked to place them in such a way that the number of crossing points is as large as possible (see Figure 1 for an example). Note that the minimal number of crossing points is 0 if the lines are parallel to each other. Obviously, the parameter n is the number of lines. For $n = 1$ there is no crossing point. For $n = 2$ there is exactly one crossing point if the lines are not parallel. The strategy is to place the n -th line in such a way that it crosses all $n - 1$ already placed lines in new $n - 1$ crossing points. This allows us to count the maximal number of available crossing points of n lines. It is

$$1 + 2 + 3 + \cdots + (n - 1),$$

and can be also derived by using the recurrence $L(n) = L(n - 1) + (n - 1)$.

Using the “small Gauss” one can get an explicit formula. But we can already use the sum above to develop an algorithm (program) counting the maximal number of crossing points of n lines.

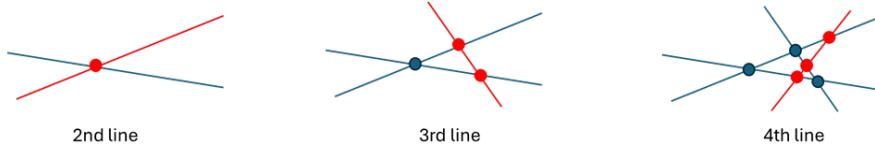


Figure 1: Crossing points when lines are added in the two-dimensional Euclidean space

The introductory example of Poya. The previous problem is only a preparation of the following problem used by Poya when introducing the power of induction. There are n lines in two-dimensional Euclidean place. What is the maximal number of areas in which the space can be partitioned in this way?

One can start with 1 line getting 2 areas. Taking 2 lines one can get 4. Taking 3 lines we will get 7 areas. Clearly, the solution cannot be 2^n , and the task starts to be challenging. For finding the solution, solving the previous task can be helpful. With the n -th line added to the previous $n - 1$ lines we can get $n - 1$ new crossing points. Each of the $n - 2$ segments between two new crossing points partitions an area into two subareas. The segment “before” the first crossing point and after the “last” crossing point also partitions areas into two subareas. Hence, adding the n -th line to the already placed $n - 1$ lines increases the number of areas by n . Therefore, the maximal number of areas obtained by placing n lines is

$$2 + 2 + 3 + \cdots + n .$$

To see this, denote the number of areas for n lines by $T(n)$ and consider $T(1) = 2$ and the derived recursion $T(n) = T(n - 1) + n$.

If you are searching for a challenge, consider the number of subareas of three-dimensional Euclidean space obtained by placing two-dimensional planes. As an exercise for students, you can take circuits instead of lines (or other suitable geometric objects) and ask to solve the problem by constructive induction.

Number of triangles in a pyramid. One can build a pyramid from equilateral triangles. In Figure 2 we see the four smallest pyramids with the heights 1, 2, 3, and 4. The question is how many triangles are in the pyramid of height n for any positive integer n . After previous experience, the students can compute $PN(1) = 1$, $PN(2) = 4$, and finally the recurrence $PN(n) = PN(n - 1) + 2n - 1$. To make the task easier the teacher may first ask for the number $A(n)$ of triangles in the lowest level of the pyramid of height n . Here one can easily establish $A(1) = 1$ and $A(n) = A(n - 1) + 2$, and so $A(n) = 2n - 1$. Then the formula $PN(n) = PN(n - 1) + A(n)$ follows. If you decide to search for an explicit formula,

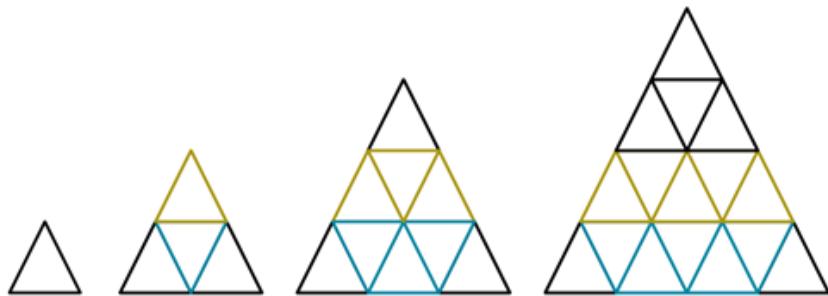


Figure 2: Pyramid built from equilateral triangles

you can derive $PN(n) = n^2$. We call attention to this because in this way you get the classical example of the sum of the first n odd numbers presented by Francesco Maurolico.

An interesting point here is that one can calculate the number of triangles in a pyramid by dividing the area of the pyramid by the area of the triangle (taking, for instance, 1 as the size of the equilateral triangle as the basic building stone). One can generate similar tasks by building pyramids from squares or hexagons.

Coloring regions of a map. The famous *four color theorem* states that 4 colors are always sufficient to color a two-dimensional map consisting of regions in such a way that no two neighboring regions have the same color. Two regions are considered to be neighbors if they have a common continuous border consisting of infinitely many points. One could ask whether fewer colors are sufficient if the partition of the map into regions is done in some regular way. Using our experience with the task from Poya, we can pose the following question: A map is partitioned into regions by n lines. How many colors are sufficient to color the map?

If you start with 1, 2, 3, or 4 lines (see [Figure 3](#)), after some attempts the students can discover that for small parameter values two colors are sufficient. So we have a hypothesis, but the question is how to prove it: The natural way is by constructive induction. First, observe that exchanging the two colors leads to a valid coloring. Secondly, add a new line to a map colored by two colors (see [Figure 4](#)). Again, we can view the new line as a sequence of segments between the crossing points with other lines. All these segments have the same colors on both sides now and all other boundaries between two regions have different colors. Keep the coloring on one side of the new line and flip the colors on the other side. In this way we get a valid coloring of the new map by two colors.

The strength of this task is that you can generate plenty of similar ones to train the class. You can take circles to partition the plane into regions, or triangles

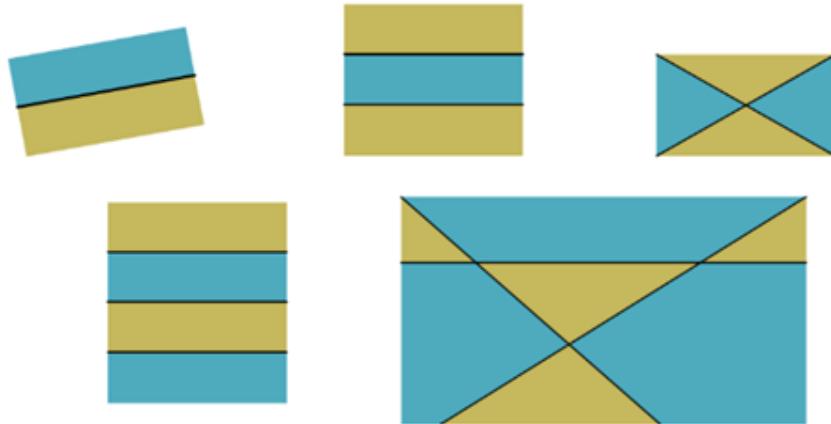


Figure 3: Two colors are sufficient to color all areas created by 1, 2, or 3 lines

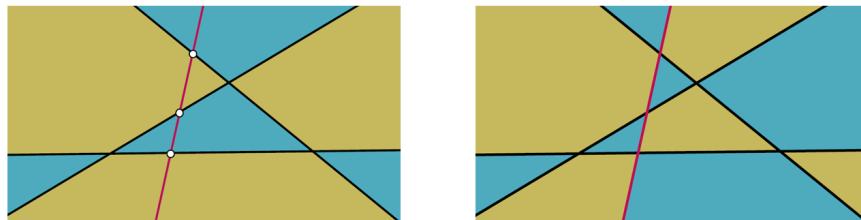


Figure 4: Adding a 4-th line and changing the color of all areas on one side of the new line

(rectangles, etc.) with and without the request that these geometrical objects intersect in a finite number of points only. If the request is satisfied, two colors are always sufficient. The strategy is to put the next object into the plane, and then flip the colors either inside of the new object or outside. If the constraint is not satisfied, one can ask students to construct examples that need at least 4 colors to be properly colored. Then the students can be asked to find the reason why the previous strategy with flipping colors inside of the new object does not work.

4 Designing Algorithms by Constructive Induction

Here we can start with the last task of the previous chapter, which is a good example for moving from problem solving to algorithm design.

Coloring regions of a map. The approach to solve the problem of dividing a plane by lines (circles, etc.) into regions offers an algorithm for two-coloring. Place the first line and color the plane with 2 colors. Then add one line after another and always exchange the colors on one side of the line.

Discovering the multiplication algorithm. Only few people are aware of the fact that the multiplication algorithm currently used in schools was designed by constructive induction. If you want to compute the product $a \cdot b$ you can parameterize by the length of the representation of b . The length of a does not matter. Suppose b consists of $n + 1$ digits and assume we can multiply by b with n digits. Let

$$b = b_n b_{n-1} \dots b_1 b_0 .$$

We can then write

$$a \cdot b = a \cdot (b_n b_{n-1} \dots b_1 b_0) = a \cdot b_n b_{n-1} \dots b_1 \cdot 10 + a \cdot b_0 .$$

So we see that one multiplication by a number consisting of n digits, one shift by one position, one multiplication by one digit, and one addition are sufficient to compute the multiplication by a number consisting of $n + 1$ digits. This is exactly the base of our school multiplication algorithm.

To train this concept one can multiply in other number systems or take other arithmetic operations [7].

Horner's method. The development of Horner's schema for evaluating a polynomial is a show case of applying constructive induction. Better than by any formula, the algorithm is explained by [Figure 5](#).

Who is the agent? We have n people and want to discover who of them is an agent. We know that there is an agent among those n people. How to recognize her or him? The agent is the person who knows everybody (all other $n - 1$ people), but nobody knows her or him. We are allowed to pose the following questions: “Person A, do you know person B?” and will always get the correct answer. The task is to find the agent with as few questions as possible.

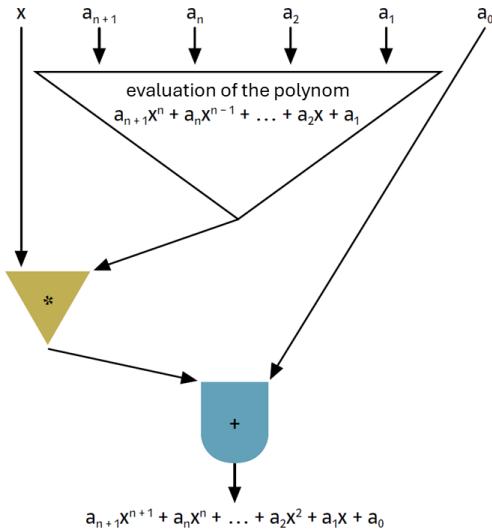


Figure 5: Horner's schema for the evaluation of a polynomial

First we observe that there can be at most one agent in the group of people. If there would be two, then each of them would know the other, and so none of them could be an agent. The induction base for the group size of one person is simple, no question is needed. For the induction step, the point is to recognize that one question is sufficient to reduce the number of candidates by 1. If we ask whether A knows B , then the answer “yes” excludes B as an agent candidate. If the answer is “no,” then A cannot be the agent. In this way we see that $n - 1$ questions are sufficient to find the agent if one knows in advance that there is an agent in the given group of people.

There is a wonderful extension of this task by allowing some restricted cheating (one or more wrong answers). This extension leads to the development of self-verifying codes for the corresponding numbers of errors.

Sorting and searching. Also in this fundamental area of algorithm design the concept of constructive induction is very fruitful. One can start with searching the minimum or maximum of n elements and design an algorithm with $n - 1$ comparisons by constructive induction.

The next step could be to do *binary search*. One can take the length of the number representation of n (approximately the discrete logarithm of n) as the parameter for the size of sorted sequences of n elements. The induction step then

shows that, if one can find an element in a sorted sequence of m elements, then one more comparison is sufficient to find an element in a sorted sequence of $2m$ elements.

For sorting algorithms one can consider *insertion sort*. If one has a sorted sequence of n elements and takes a new element, then one has to find its position in the sorted sequence. One can get different algorithms depending on the strategy used. If one uses *bubble sort* for placing the new element, the resulting algorithm has a quadratic number of comparisons. If one uses *binary search*, the complexity is in $O(n \log n)$.

Another sorting algorithm can search for the maximum (as bubble sort does) and then sort the remaining $n - 1$ elements.

There are plenty of possibilities to create further exercises here. For instance, one can search for the k largest elements.

Winning strategies for games. Constructive induction is a genius strategy for searching for winning strategies for finite games. You start with your winning configurations, and in each constructive step you label the winning configurations until all winning configurations are labeled. You do the same for the losing configurations. The labeling rules are very simple: Start with your winning configurations and losing configurations defined by the game, and in each constructive step you label further your winning or losing configurations.

You continue until all configurations of the game are labelled. You label a configuration as your winning configuration if you can move from this configuration in one step to a configuration that is already labeled as your winning configuration (if the turn is yours). If it's your opponent's turn, you label a configuration as your winning configuration if she or he can only move to one of your winning configurations. When it is your turn, you label a configuration as your losing configuration (winning configuration of your contrary) if all your possible moves end in an already marked winning configuration of your opponent. Also, you label a configuration as your losing configuration, if your opponent can reach in one step a winning configuration for her or him.

There are many simple games that can be completely analyzed by this strategy [7].

Dijkstra's algorithm. This is an advanced example. Especially since the number of different paths between two vertices in a network can be exponential in the size of the network, and we want to avoid looking at all possible paths in order to find the shortest one.

The key issue when developing Dijkstra's algorithm by constructive induction is the choice of the parameter. First, one defines the problem of finding the shortest

paths in a network from the source to the k closest vertices. Then k is the parameter, not the size of the network. The base for $k = 1$ is easy, one takes that neighbor of the source that is connected to it by the cheapest edge. The induction step takes the tree with the k shortest paths to the k closest vertices and argues that the shortest path to the $(k + 1)$ -th closest vertex must go via the edges of the tree of the k shortest paths. Then one can efficiently estimate the next closest vertex by considering only edges of the tree and the edges leading from the tree to the vertices not belonging to the k closest ones.

If one wants to start teaching or discovering Dijkstra's algorithm with an easier task, then one can search for shortest paths from a source to all other vertices in a network whose edges all have the same value and so develop the *breadth-first-search* algorithm.

5 Conclusion

If one wants to strengthen students in algorithmic (computational) thinking, one should not try to correctly execute presented algorithms only. Instead, one has to push the students to use and develop abstractions and problem solving competencies. This requires developing robust strategies for designing algorithms for a large variety of problems. Teaching in high school asks for natural and well understandable strategies.

Constructive induction is the oldest robust strategy for solving problems and it is very transparent and comprehensible to students. But the key point is that constructive induction has a very high educational value with respect to the development of the student's way of thinking. Except the potential of introducing induction as a proof method, constructive induction offers a special case of recursion and so can be used as an easy introduction to the recursive algorithm design method "divide et impera" (divide and conquer). The induction step results in a recurrence that corresponds to the reduction of solving problem instances of size n to solving problem instances of size smaller than n , and so to the concept of "divide et impera." Also note that using constructive induction in problem solving consequently from smaller instance sizes to larger ones is also the base of "dynamic programming," another fundamental algorithm design technique.

In this paper we only outlined what kind of problems can be approached by constructive induction in high schools. A careful implementation of teaching solving problems by constructive induction is presented in our textbook [7], which shows how to guide students such that they discover solutions and develop algorithms to a large extend on their own.

References

- [1] J. Arnold, C. Donner, U. Hauser, M. Hauswirth, J. Hromkovič, T. Kohn, D. Komm, D. Maletinsky, and N. Roth (2019). *Programmieren und Robotik*. Klett und Balmer AG, Baar, Switzerland.
- [2] B. du Boulay (1986). Some difficulties of learning to program. *Journal of Educational Computing Research* 2(1):57–73.
- [3] W. H. Bussey (1917). The origin of mathematical induction. *The American Mathematical Monthly* 24(5):199–207.
- [4] A. Cypher (1993). *Watch What I Do: Programming by Demonstration*. MIT Press. Cambridge, MA, USA and London, UK.
- [5] Encyclopedia of Mathematics (2021). *Mathematical Induction*. Online: http://encyclopediaofmath.org/index.php?title=Mathematical_induction, last accessed September 20, 2024.
- [6] S. Fincher, J. Jeuring, C. S. Miller, P. Donaldson, B. du Boulay, M. Hauswirth, A. Hellas, F. Hermans, C. Lewis, A. Mühling, J. L. Pearce, and A. Petersen (2020). Notional machines in computing education: the education of attention. In *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education (ITiCSE-WGR 2020)*. ACM, New York, NY, USA, pp. 21–50.
- [7] J. Gallenbacher, J. Hromkovič, R. Lacher, D. Komm, and H. Pierhöfer (2023). *Algorithmen und Künstliche Intelligenz*. Klett and Balmer AG, Baar, Switzerland.
- [8] J. Hromkovič, T. Kohn, D. Komm, and G. Serafini (2016). Combining the power of Python with the simplicity of Logo for a sustainable computer science education. In *Proceedings of the 9th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives (ISSEP 2016)*. LNCS 9973, pp. 155–166. Springer.
- [9] J. Hromkovič and R. Lacher (2023). How teaching informatics can contribute to improving education in general. *Bulletin of EATCS* 139.
- [10] T. Kohn (2017). *Teaching Python Programming to Novices: Addressing Misconceptions and Creating a Development Environment*. Dissertation 24076, ETH Zürich, Switzerland.
- [11] T. Kohn and D. Komm (2018). Teaching programming and algorithmic complexity with tangible machines. In *Proceedings of Informatics in Schools: Fundamentals of Computer Science and Software Engineering (ISSEP 2018)*. LNCS 11169, pp. 68–83. Springer.
- [12] H. Lieberman (2001). *Your Wish is My Command: Programming By Example*. Morgan Kaufmann, San Francisco, CA, USA.
- [13] S. Papert (1980). *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books, Inc., New York, NY, USA.
- [14] G. Vacca (1909). Maurolycus, the first discoverer of the principle of mathematical induction. *Bulletin of the American Mathematical Society* 16(2):70–73.

The Bulletin of the EATCS

The Bulletin of the EATCS

THE LOGIC IN COMPUTER SCIENCE COLUMN

BY

YURI GUREVICH

Computer Science & Engineering
University of Michigan, Ann Arbor, Michigan, USA
gurevich@umich.edu

ON A MEASURE OF INTELLIGENCE

Yuri Gurevich

The measure of intelligence is the ability to change.

— Albert Einstein¹

Abstract

The Fall 2024 column is a little discussion on intelligence, measuring intelligence, and related issues, provoked by a fascinating must-read article “On the measure of intelligence” by François Chollet. The discussion includes a modicum of critique of the article.

§1 Cybernetics vs. AI, and podcasts vs. reading

Quisani² (walking in): What are you reading?

Author: An article “On the measure of intelligence” by François Chollet [3].

Q: Is it about psychology?

A: It is mostly about AI. Chollet is a prominent figure in AI.

Q: We spoke about AI last spring. But you didn’t seem to be interested in AI before that.

A: This is largely correct, though I read Norbert Wiener’s “Cybernetics” [18], when it was translated to Russian in 1968, and was taken with it. For a while I tried to follow cybernetics developments, at least in the USSR.

Q: What’s cybernetics?

A: Wiener gives a concise definition in the subtitle of that book of his: “Control and communication in the animal and the machine.”

Q: How is this different from AI?

¹The saying is commonly attributed to Einstein. A similar saying “Intelligence is the ability to adapt to change” is commonly attributed to Steven Hawking. In neither case is there a clear reference.

²My former student

A: This is a good question. Here is an explanation by Michael Jordan, UC Berkeley Professor, not the basketball player:

It was John McCarthy (while a professor at Dartmouth, and soon to take a position at MIT) who coined the term AI, apparently to distinguish his budding research agenda from that of Norbert Wiener (then an older professor at MIT). Wiener had coined “cybernetics” to refer to his own vision of intelligent systems — a vision that was closely tied to operations research, statistics, pattern recognition, information theory, and control theory. McCarthy, on the other hand, emphasized the ties to logic. In an interesting reversal, it is Wiener’s intellectual agenda that has come to dominate in the current era, under the banner of McCarthy’s terminology [8].

In his conversation with Lex Fridman, Jordan tells this story in more colorful terms [9, 18:06].

In the USSR of my time (I left “the land of victorious socialism” in 1973) and for a long time after, the term cybernetics was used for the field.

Q: Do you spend much time reading stuff outside your immediate research interests?

A: I do, though these days I spend more time listening to podcasts.

Q: Why?

A: You can walk and listen to a podcast; you can’t walk and read.

Q: I can’t but some do, even as they cross the road. But I digress.

A: A podcast often provokes reading. Something attracts your attention, and you want to know more about it. That is exactly what happened in this case. First I heard François Chollet on a podcast [4].

§2 The g-factor

Q: Is Chollet’s article about IQ scores? I have never heard of any other measure of intelligence.

A: Chollet mentions IQ scores and the g-factor, but his 64-page article goes far beyond that.

Q: What’s the g-factor?

BEATCS no 144

A: British psychologist Charles Spearman noticed positive correlations when a person performs different tests. He suggested that general intelligence (g-factor) is a single underlying ability influencing (but not determining) performance across different cognitive tasks [13].

Q: I know people who perform well on one kind of problems, say complex word problems, but not so well on another kind of problems, say mathematical problems.

A: Richard Haier, the editor-in-chief of *Intelligence*, says that, typically, the correlation is still positive [6, 04:32].

Q: How well has Spearman's idea held up?

A: There is an ongoing debate about its validity and limitations, especially in modern psychology, but it seems to be accepted in principle. "I think I can say without fear of being empirically contradicted that it is the most replicated finding in all of psychology" [6, 06:05].

Q: Wow! And IQ scores are supposed to measure the g-factor?

A: Modern IQ tests are designed to do just that by assessing various cognitive domains.

Q: It is hard to believe that intelligence, that is general intelligence, can be summarized by one number.

A: I agree with you. Intelligence is too complex and multifaceted to be reduced to one number. But the proponents of IQ scores point out its empirical support, practicality, and predictive utility. This issue is too involved and important to be discussed on one foot, as they say in Hebrew. We would need to give it the time and attention it deserves.

Q: I understand.

§3 Intelligence as skill-acquisition efficiency

Q: I guess Chollet also wants to measure the g-factor. What is special about his approach?

A: To make progress towards the promise of AI, one needs a workable definition of intelligence and a quantitative measure of intelligence – in particular human-like general intelligence. Chollet makes a good step in this direction.

We ... articulate a new formal definition of intelligence based on Algorithmic Information Theory, describing intelligence as *skill-acquisition efficiency* and highlighting the concepts of *scope, generalization difficulty, priors, and experience*, as critical pieces to be accounted for in characterizing intelligent systems [3, Abstract].

A task-specific performance, say at chess or Go, measures a particular skill; the performance can be inflated through training data and prior knowledge. Whether you are a human or an AI system, general intelligence is not about what you know or can do. According to Chollet, it is about how efficiently you can acquire a new skill or adapt to a new environment that you did not anticipate. “There’s a big distinction to be drawn between intelligence, which is a process, and the output of that process which is skill” [2, 28:52].

Q: I always thought about intelligence as a capacity, rather than a process. Maybe Chollet means “skill-acquisition” which is a process. In any case, I like very much the definition of intelligence as skill-acquisition efficiency. What about you?

A: I doubt that the sprawling intuitive notion of intelligence reduces to skill-acquisition efficiency but it is hard to argue with the thesis that skill-acquisition efficiency is a manifestation of intelligence.

Q: What does efficiency mean in Chollet’s definition? Humans can often grasp a new concept with just a few examples. Is that the kind of efficiency he is talking about?

A: Yes, an efficient system can generalize from limited examples to solve a broader range of related problems. This is an important aspect of skill-acquisition efficiency. But there are other aspects. In particular, an efficient system can quickly adapt to moderate changes in its environment without extensive retraining, can apply knowledge from previously learned tasks to a new one, and can use less compute.

§4 Algorithmic Information Theory (AIT)

Q: It is surprising that skill-acquisition efficiency has a formal definition. Does Chollet prove interesting theorems about it?

A: Chollet expresses various notions involved in his definition of intelligence in the AIT formalism. He proves no theorems. To me, the formalization exercise is not convincing. It is not even sound in a sense.

Q: What do you mean?

BEATCS no 144

A: Let me introduce Chollet's notation and, at the same time, recall algorithmic complexity, also known as Kolmogorov complexity [11, 12]. Below, strings are binary strings.

The Algorithmic Complexity (noted $H(s)$) of a string s is ... the length of the shortest program that outputs the string when running on a fixed universal Turing machine. Since any universal Turing machine can emulate any other universal Turing machine, $H(s)$ is machine-independent to a constant. ... [D]efine the information content that a string s_2 possesses about a string s_1 (called "Relative Algorithmic Complexity" and noted $H(s_1|s_2)$) as the length of the shortest program that, taking s_2 as input, produces s_1 [3, p. 34].

Machine-independence follows from the fact that a given universal programming language L can emulate any universal programming language L' (as well as any non-universal programming language) up to an additive constant. As a result, $H_L(s) \leq H_{L'}(s) + c$ and $H_L(s_1|s_2) \leq H_{L'}(s_1|s_2) + c$, where c is essentially the length of an L' -to- L translator. If machine independence is a must, then Chollet has a problem. For example, he formalizes the *generalization difficulty* $\text{GD}_{T,C}^\theta$ of a task T given an experience curriculum C as

$$\frac{H(\text{Sol}_T^\theta | \text{TrainSol}_{T,C}^{\text{opt}})}{H(\text{Sol}_T^\theta)} \quad (\text{GD})$$

where Sol_T^θ is the shortest solution of T of threshold θ during evaluation, and $\text{TrainSol}_{T,C}^{\text{opt}}$ is the shortest optimal training-time solution of T given C , so that $0 < \text{GD}_{T,C}^\theta \leq 1$. (Chollet describes $T, C, \theta, \text{Sol}_T^\theta$, and $\text{TrainSol}_{T,C}^{\text{opt}}$ more precisely, but this is not important for our purposes.) The allegedly innocent additive constant makes $\text{GD}_{T,C}^\theta$ highly dependent on the fixed universal language, not just up to an additive constant.

Q: Explain.

A: Notice that the additive constant c may be arbitrarily large. Indeed, flip a fair coin N times, thus producing a random string s of length N . Then, with overwhelming probability, $H(s) \approx N$ for the fixed language L [12, p. 8]. If L' is the extension of L with a single short command that prints s then, for L' , $H(s)$ is a small constant, and so $c \approx N$.

Let A and B be the numerator and denominator in (GD). Consider a situation where, for our fixed language L , $\text{GD}_{T,C}^\theta$ is small, say $A = 1$ and $B = 100$, so that $\text{GD}_{T,C}^\theta = 1/100$. Let L' be another language whose Kolmogorov complexities exceed those of L by 1000000. Then, for L' , we have $A = 1000001$ and $B = 1000100$, so that $\text{GD}_{T,C}^\theta \approx 1$, so that $\text{GD}_{T,C}^\theta$ moves from one extreme (≈ 0) to the other (≈ 1).

Q: What does curriculum mean in Chollet's approach?

A: A *curriculum* is a (carefully crafted) sequence of tasks. It is an important concept in Chollet's approach. Curricula aim to cover a wide range of task-solving skills and are designed to minimize the reliance on pre-existing knowledge.

Q: Back to your critique, while machine independence is desirable, one still can work with a fixed universal programming language, carefully chosen to minimize built-in information.

A: This is true, at least in principle. But let me notice that the functions $H(s)$ and $H(s_1|s_2)$ are uncomputable. Moreover, they are not even approximable by computable functions. Indeed suppose toward a contradiction that $|H(s) - f(s)| \leq N$ for some computable $f(s)$. Then $f(s) - N$ is a computable lower bound for $H(s)$. By Theorem 6 in [12], $f(s) - N$ is bounded. Then $f(s)$ is bounded and therefore $H(s)$ is bounded, which is impossible.

Q: Still, there may be something useful in algorithmic complexity. An expression like (GD) is more succinct than a textual description of it.

A: Yes, symbolic expressions may be succinct, and this is an advantage.

Q: How important is the AIT foundation for Chollet's approach?

A: I don't think it is important. Chollet's approach isn't really founded in AIT. In the literature, AIT is typically used as motivation and inspiration rather than literally, and it can play such a role in Chollet's approach as well.

§5 Measuring intelligence

Q: What else is there in Chollet's article?

A: He covers a lot of ground. There is a general theme of defining intelligence, that we discussed above, and measuring intelligence, that I intend to bring up. But many sections are educational essays in their own right, at least for non-experts. For example, there is a section §I.3.2 on generalization theory. It addresses defining, measuring, and maximizing generalization.

Q: Advanced mathematics is all about generalizations. But I have never heard of measuring generalization. Maximizing generalization would probably result in some trivial scenario, with no interesting theorems.

A: Chollet speaks about different notion of generalization, "originally developed to characterize how well a statistical model performs on inputs that were not part of its training data" [3, p. 9].

BEATCS no 144

Measuring intelligence is super important to Chollet: “we need to be able to define and evaluate intelligence in a way that enables comparisons between two [AI] systems, as well as comparisons with humans” [3, Abstract]. To this end, he reifies his definition of intelligence as “skill-acquisition efficiency over a scope of tasks, with respect to priors, experience, and generalization difficulty” [3, p. 27].

Q: I have encountered the noun prior before, but only in Bayesian statistics, where it means prior distribution.

A: In AI the noun prior, often in the form priors, is used to mean previously acquired knowledge that the agent, an AI system or a human, brings to the table. Chollet proposes a new benchmark for general intelligence that he calls Abstraction and Reasoning Corpus (ARC).

Q: How does ARC differ from previous benchmarks?

A: Traditionally, AI benchmarks focused on tasks solvable through pattern recognition and statistical learning. ARC emphasizes generalization, abstraction, and few-shot learning.

Q: I haven’t heard of few-shot learning but the name seems to suggest learning new tasks from a small number of examples.

A: Exactly.

Q: ARC seems ambitious.

A: It is very ambitious, a paradigm shift in how we assess intelligence. ARC aims to provide a standard measure for generalization and abstraction abilities of different intelligent systems, whether AI systems or humans.

Q: Is ARC already available? Are there some tests that I can try?

A: Yes, see [1]. There are quite a number of tests.

Q: Great. I intend to try some of those tests, at least on myself.

A: You will need a GitHub account and a bit of programming prowess to make those tests available in convenient visual form.

Q: ARC looks super attractive but also super challenging, especially if one wants to compare AI systems with humans.

A: I agree. The project is grandiose, and challenges are abundant. In particular, ARC involves new algorithms to assess abstract reasoning and few-shot learning.

Q: Does Chollet propose ARC as an adequate measure of intelligence?

A: Not at this point. ARC is a project in progress. The current implementation presupposes only human core knowledge and even that only in part. One has to start somewhere.

Q: What's core knowledge?

A: The knowledge that we are born with or hardwired to acquire quickly after birth.

Q: To compare AI systems with humans, Chollet needs to make human core knowledge explicit. Is this even possible?

A: There has been impressive progress on understanding the human core knowledge [14]. The core knowledge comprises a few distinct systems, i.e. distinct priors. One prior allows us to see the world split into distinct objects. Another prior is that some of these objects are goal-pursuing agents. Two additional priors are related to primitive geometry and numbering respectively. “Human cognition is founded, in part, on four systems for representing objects, actions, number, and space. It may be based, as well, on a fifth system for representing social partners” [14, Abstract].

§6 Reaction to Chollet’s paper

Q: Has Chollet’s definition of intelligence been accepted?

A: It generated significant debate in the AI research community. It influenced discussions on AI evaluation metrics and benchmarks. It sparked renewed interest in developing more comprehensive intelligence tests for AI systems. But it’s not universally accepted. Some question whether Chollet’s definition of intelligence is comprehensive. Does it capture creativity? Does it capture emotional intelligence?

Q: I understand creativity doubts. As far as emotional intelligence is concerned, my impression is that it is irrelevant here. That is unless advanced intelligence is impossible without emotional intelligence.

A: Recent advances in neuroscience shed light on this question. Jeff Hawkins, neuroscientist and businessman, addresses the issue head-on in his beautiful book *The Thousand Brains Theory of Intelligence* [7].

The parts of the brain responsible for intelligence and emotion are distinct, albeit connected.

The newest part of our brain is the neocortex . . . The neocortex is the organ of intelligence. . . . Fears and emotions are created by neurons in the old brain [7, §1].

Emotionless artificial intelligence is not only possible; it is more straightforward.

Intelligent machines need to have a model of the world and the flexibility of behavior that comes from that model, but they don't need to have human-like instincts for survival and procreation [7, §10]....

We, the designers of intelligent machines, have to go out of our way to design in motivations [7, §11].

Q: Should we design in emotions?

A: Personally, I don't think so. Can you switch off a computer that implores you not to do that? But the issue is deep.

Q: I am becoming a serial digresser ☺ Let's return to Chollet. Since he has the ambition to compare AI with humans, what do neuroscientists say? What, if anything, do computer scientists say?

A: I don't have a representative sample of opinions. My impression is that, by and large, neuroscientists thought and continue to think that intelligence is poorly defined. Recall Michael Jordan whom I quoted earlier. He says: "We don't know what intelligence is. We don't know much about abstraction and reasoning at the level of humans; we don't have a clue" [9, 16:28].

Computer scientist Leslie Valiant, who received the Turing Award in 2010, has a similar opinion, with a twist. In his conversation with Sean Carroll [16] he says:

I think the main downside of intelligence is that no one can define it. That is,

of course, people have complained that we give importance to intelligence.

We test people for intelligence, and this has consequences. And we don't even know what we're testing for, where the questions come from. So I think it's very unfortunate that the notion of intelligence has become so important, because it's not explicitly defined. So I've explicitly defined educability; intelligence has no such definition [14, 45:51].

Q: I see what you mean by the twist. Tell me about educability.

A: I will, though not now. Valiant's approach [13] deserves a deep dive, a separate conversation.

Q: OK, I understand. I wonder whether Michael Jordan and Leslie Valiant heard of Chollet's definition of intelligence and what they think about it.

A: I asked Valiant about that. Here's his reply:

I define a model (educability) and describe what real world phenomenon it is intended to correspond to (extra cognitive capabilities of humans). Chollet does indeed also have a model. But my question is: what real world phenomenon does it correspond to? Intelligence as generally used is a broad sprawling concept but some aspects, such as what IQ tests measure, seem inseparable from it. I think it is difficult to argue that any model of intelligence corresponds well to the intuitive concept as widely used. I think the challenge for any model is to articulate what important phenomenon it captures [15].

§7 Logic

Q: We can't finish without touching on logic. I am looking at the official site of Abstraction and Reasoning Corpus for Artificial General Intelligence [1]. The terms abstraction and reasoning should make the logician in you happy.

A: You bet. But such abstractions and reasoning are a challenge to the logic community.

Q: Yes, we spoke about this last spring [5].

A: Let me add a few words. Many mathematicians and mathematical logicians look down on philosophers. There is a joke about the chairman of math department trying to squeeze out a new position from the dean:

- Contrary to physicists and chemists, mathematicians don't need labs, just a pen, paper, and a waste basket.
- You think you are cheap. The chairman of the philosophy department tells me that philosophers need only a pen and paper, no waste basket.

Q: That's cruel. Also, philosophical departments don't have a monopoly on philosophy.

A: I agree on both counts. In any case, mathematical logicians should not forget that it was essentially philosophical work of formalization that prepared the ground for mathematical logic. Think of Boole, Cantor, Frege, Russell & Whitehead, Zermelo. The logics of today are not sufficient to satisfy the needs of rapidly developing AI.

Q: Yes, in our conversation last spring, you mentioned fast thinking in the sense of Kahneman [10].

A: Right. We need new foundational investigations, new formalizations which would enable us to develop logics appropriate for new applications.

Q: Who will do the work? Mathematical logicians or philosophers? Or maybe computer scientists?

A: The work requires (at least) intimate knowledge of AI, mathematical maturity, and a philosophical/foundational attitude. An appropriate person can start his career in AI, computer science, logic, mathematics, philosophy, or some different field altogether.

Q: It could be a team.

A: Yes, it could be a team. Russell and Whitehead worked as a team, with a clear division of labor. Daniel Kahneman and Amos Tversky did their foundational work, albeit on psychology rather than logic, as a closely knit team [10]

Acknowledgments

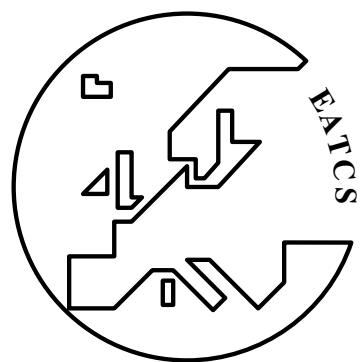
Many thanks to Andreas Blass and Alexander Shen for generously and repeatedly commenting on drafts of the paper.

References

- [1] Abstraction and Reasoning Corpus for Artificial General Intelligence (ARC-AGI) official site, <https://github.com/fchollet/ARC-AGI>
- [2] François Chollet, “Keras, Deep Learning, and the Progress of AI,” *Lex Fridman Podcast* #38, 14 September 2019, <https://youtu.be/Bo8MY4JpiXE>
- [3] François Chollet, “On the measure of intelligence,” arXiv:1911.01547v2
- [4] François Chollet, “Measures of Intelligence,” *Lex Fridman Podcast* #120, 30 August 2020, <https://youtu.be/PUAdj3w3w04>
- [5] Yuri Gurevich, “Logic and Generative AI,” *Bulletin of EATCS* 143, June 2024
- [6] Richard J. Haier, “IQ Tests, Human Intelligence, and Group Differences,” *Lex Fridman Podcast* #302, 14 July 2022, <https://youtu.be/hppbxV9C63g>
- [7] Jeff Hawkins, “The Thousand Brains: A New Theory of Intelligence,” Basic Books, New York, 2021
- [8] Michael I. Jordan, “Artificial Intelligence—The Revolution Hasn’t Happened Yet,” *Harvard Data Science Review* 1.1, Summer 2019, <https://hdsr.mitpress.mit.edu/pub/wot7mkc1/release/10>
- [9] Michael I. Jordan, “Machine Learning, Recommender Systems, and Future of AI,” *Lex Fridman Podcast* #74, 24 February 2020, https://youtu.be/EYIKy_FM9x0
- [10] Daniel Kahneman, “Thinking fast and slow,” Farrar, Straus and Giroux 2011, <https://youtu.be/UwwBG-Mbn1Y>
- [11] Ming Li and Paul Vitányi, “An Introduction to Kolmogorov Complexity and its Application,” 3rd edition, Springer 2008
- [12] Alexander Shen, Vladimir A. Uspensky, and Nikolay K. Vereshchagin, “Kolmogorov Complexity and Algorithmic Randomness,” American Mathematical Society 2017
- [13] Charles Spearman, “‘General Intelligence,’ Objectively Determined and Measured,” *The American Journal of Psychology* 15:2 (1904) 201–292
- [14] Elizabeth S. Spelke and Katherine D. Kinzler, “Core Knowledge,” *Developmental Science* 10:1 2007 89—96
- [15] Leslie Valiant, “The Importance of Being Educable,” Princeton University Press 2024
- [16] Leslie Valiant, “Learning and Educability in Computers and People,” *Mindscape* 272, Sean Carroll’s Podcast, 15 April 2024, <https://youtu.be/FHW-nBIZc2g>
- [17] Leslie Valiant, private communication, August 2024
- [18] Norbert Wiener, “Cybernetics: Control and Communication in the Animal and the Machine,” Second edition, MIT Press, 1961

BEATCS no 144

European
Asso**c**i**a**n**t**ion f**o**r
Theoretical
Computer
Science



E **A** **T** **C** **S**
134

EATCS

HISTORY AND ORGANIZATION

EATCS is an international organization founded in 1972. Its aim is to facilitate the exchange of ideas and results among theoretical computer scientists as well as to stimulate cooperation between the theoretical and the practical community in computer science.

Its activities are coordinated by the Council of EATCS, which elects a President, Vice Presidents, and a Treasurer. Policy guidelines are determined by the Council and the General Assembly of EATCS. This assembly is scheduled to take place during the annual International Colloquium on Automata, Languages and Programming (ICALP), the conference of EATCS.

MAJOR ACTIVITIES OF EATCS

- Organization of ICALP;
- Publication of the "Bulletin of the EATCS;"
- Award of research and academic career prizes, including the EATCS Award, the Gödel Prize (with SIGACT), the Presburger Award, the EATCS Distinguished Dissertation Award, the Nerode Prize (joint with IPEC) and best papers awards at several top conferences;
- Active involvement in publications generally within theoretical computer science.

Other activities of EATCS include the sponsorship or the cooperation in the organization of various more specialized meetings in theoretical computer science. Among such meetings are: CIAC (Conference of Algorithms and Complexity), CiE (Conference of Computer Science Models of Computation in Context), DISC (International Symposium on Distributed Computing), DLT (International Conference on Developments in Language Theory), ESA (European Symposium on Algorithms), ETAPS (The European Joint Conferences on Theory and Practice of Software), LICS (Logic in Computer Science), MFCS (Mathematical Foundations of Computer Science), WADS (Algorithms and Data Structures Symposium), WoLLIC (Workshop on Logic, Language, Information and Computation), WORDS (International Conference on Words).

Benefits offered by EATCS include:

- Subscription to the "Bulletin of the EATCS;"
- Access to the Springer Reading Room;
- Reduced registration fees at various conferences;
- Reciprocity agreements with other organizations;
- 25% discount when purchasing ICALP proceedings;
- 25% discount in purchasing books from "EATCS Monographs" and "EATCS Texts;"
- Discount (about 70%) per individual annual subscription to "Theoretical Computer Science;"
- Discount (about 70%) per individual annual subscription to "Fundamenta Informaticae."

Benefits offered by EATCS to Young Researchers also include:

- Database for Phd/MSc thesis
- Job search/announcements at Young Researchers area

(1) THE ICALP CONFERENCE

ICALP is an international conference covering all aspects of theoretical computer science and now customarily taking place during the second or third week of July. Typical topics discussed during recent ICALP conferences are: computability, automata theory, formal language theory, analysis of algorithms, computational complexity, mathematical aspects of programming language definition, logic and semantics of programming languages, foundations of logic programming, theorem proving, software specification, computational geometry, data types and data structures, theory of data bases and knowledge based systems, data security, cryptography, VLSI structures, parallel and distributed computing, models of concurrency and robotics.

SITES OF ICALP MEETINGS:

- Paris, France 1972
- Saarbrücken, Germany 1974
- Edinburgh, UK 1976
- Turku, Finland 1977
- Udine, Italy 1978
- Graz, Austria 1979
- Noordwijkerhout, The Netherlands 1980
- Haifa, Israel 1981
- Aarhus, Denmark 1982
- Barcelona, Spain 1983
- Antwerp, Belgium 1984
- Nafplion, Greece 1985
- Rennes, France 1986
- Karlsruhe, Germany 1987
- Tampere, Finland 1988
- Stresa, Italy 1989
- Warwick, UK 1990
- Madrid, Spain 1991
- Wien, Austria 1992
- Lund, Sweden 1993
- Jerusalem, Israel 1994
- Szeged, Hungary 1995
- Paderborn, Germany 1996
- Bologna, Italy 1997
- Aalborg, Denmark 1998
- Prague, Czech Republic 1999
- Genève, Switzerland 2000
- Heraklion, Greece 2001
- Malaga, Spain 2002
- Eindhoven, The Netherlands 2003
- Turku, Finland 2004
- Lisabon, Portugal 2005
- Venezia, Italy 2006
- Wrocław, Poland 2007
- Reykjavik, Iceland 2008
- Rhodes, Greece 2009
- Bordeaux, France 2010
- Zürich, Switzerland 2011
- Warwick, UK 2012
- Riga, Latvia 2013
- Copenhagen, Denmark 2014
- Kyoto, Japan 2015
- Rome, Italy 2016
- Warsaw, Poland 2017
- Prague, Czech Republic 2018
- Patras, Greece 2019
- Saarbrücken, Germany (virtual conference) 2020
- Glasgow, UK (virtual conference) 2021
- Paris, France 2022
- Paderborn, Germany 2023
- Tallinn, Estonia 2024

(2) THE BULLETIN OF THE EATCS

Three issues of the Bulletin are published annually, in February, June and October respectively. The Bulletin is a medium for *rapid* publication and wide distribution of material such as:

- EATCS matters;
- Technical contributions;
- Columns;
- Surveys and tutorials;
- Reports on conferences;
- Information about the current ICALP;
- Reports on computer science departments and institutes;
- Open problems and solutions;
- Abstracts of Ph.D. theses;
- Entertainments and pictures related to computer science.

The Bulletin of the EATCS

Contributions to any of the above areas are solicited, in electronic form only according to formats, deadlines and submissions procedures illustrated at <http://www.eatcs.org/bulletin>. Questions and proposals can be addressed to the Editor by email at bulletin@eatcs.org.

(3) OTHER PUBLICATIONS

EATCS has played a major role in establishing what today are some of the most prestigious publication within theoretical computer science.

These include the *EATCS Texts* and the *EATCS Monographs* published by Springer-Verlag and launched during ICALP in 1984. The Springer series include *monographs* covering all areas of theoretical computer science, and aimed at the research community and graduate students, as well as *texts* intended mostly for the graduate level, where an undergraduate background in computer science is typically assumed.

Updated information about the series can be obtained from the publisher.

The editors of the EATCS Monographs and Texts are now M. Henzinger (Vienna), J. Hromkovič (Zürich), M. Nielsen (Aarhus), G. Rozenberg (Leiden), A. Salomaa (Turku). Potential authors should contact one of the editors.

EATCS members can purchase books from the series with 25% discount. Order should be sent to:

*Prof.Dr. G. Rozenberg, LIACS, University of Leiden,
P.O. Box 9512, 2300 RA Leiden, The Netherlands*

who acknowledges EATCS membership and forwards the order to Springer-Verlag.

The journal *Theoretical Computer Science*, founded in 1975 on the initiative of EATCS, is published by Elsevier Science Publishers. Its contents are mathematical and abstract in spirit, but it derives its motivation from practical and everyday computation. Its aim is to understand the nature of computation and, as a consequence of this understanding, provide more efficient methodologies. The Editor-in-Chief of the journal currently are D. Sannella (Edinburgh), L. Kari and P.G. Spirakis (Patras).

ADDITIONAL EATCS INFORMATION

For further information please visit <http://www.eatcs.org>, or contact the President of EATCS:

*Prof. Giuseppe F. Italiano,
Email: president@eatcs.org*

EATCS MEMBERSHIP

DUES

The dues are €40 for a period of one year (two years for students / Young Researchers). Young Researchers, after paying, have to contact secretary@eatcs.org, in order to get additional years. A new membership starts upon registration of the payment. Memberships can always be prolonged for one or more years.

In order to encourage double registration, we are offering a discount for SIGACT members, who can join EATCS for €35 per year. We also offer a five-euro discount on the EATCS membership fee to those who register both to the EATCS and to one of its chapters. Additional €35 fee is required for ensuring the *air mail* delivery of the EATCS Bulletin outside Europe.

HOW TO JOIN EATCS

You are strongly encouraged to join (or prolong your membership) directly from the EATCS website www.eatcs.org, where you will find an online registration form and the possibility of secure online payment. Alternatively, contact the Secretary Office of EATCS:

*Mrs. Efi Chita,
Computer Technology Institute & Press (CTI)
1 N. Kazantzaki Str, University of Patras campus,
26504, Rio, Greece
Email: secretary@eatcs.org,
Tel: +30 2610 960333, Fax: +30 2610 960490*

If you are an EATCS member and you wish to prolong your membership or renew the subscription you have to use the Renew Subscription form. The dues can be paid via paypal and all major credit cards are accepted.

For additional information please contact the Secretary of EATCS:

*Dmitry Chistikov
Computer Science
University of Warwick
Coventry
CV4 7AL
United Kingdom
Email: secretary@eatcs.org,*
